UNIVERSIDAD DE PIURA

FACULTAD DE INGENIERÍA

Diseño automático de un tijeral con software Matlab y su verificación con el software de elementos finitos en Solidworks

Tesis para optar el Título de
Ingeniero Mecánico - Eléctrico

Diego Angel Ferre Fenco
Franco Augusto Ramirez Cardoza

Asesor(es):
Dr. Ing. Miguel Buenaventura Castro Sánchez

Piura, diciembre de 2023

UNIVERSIDAD DE PIURA

Declaración Jurada de Originalidad del Trabajo Final

Yo, Diego Angel Ferre Fenco, egresado del Programa Académico de Ingeniería Mecánico-Eléctrica de la Facultad de Ingeniería de la Universidad de Piura, identificado(a) con DNI N° 72517351.

Declaro bajo juramento que:

1. Soy autor del trabajo final titulado:
   "Diseño automático de un tijeral con software Matlab y su verificación con el software de elementos finitos en Solidworks. "
   El mismo que presento bajo la modalidad de Tesis[1] para optar el Título profesional[2] de Ingeniero Mecánico - Eléctrico.

2. Que el trabajo se realizó en coautoría con los siguientes alumnos de la Universidad de Piura.
   - Franco Augusto Ramirez Cardoza, identificado con DNI N° 75142008
3. La asesoría del trabajo estuvo a cargo de:
   - Dr. Ing. Miguel Buenaventura Castro Sánchez, identificado con DNI N° 02821943
4. El texto de mi trabajo final respeta y no vulnera los derechos de terceros o de ser el caso derechos de los coautores, incluidos los derechos de propiedad intelectual, datos personales, entre otros. En tal sentido, el texto de mi trabajo final no ha sido plagiado total ni parcialmente, para la cual he respetado las normas internacionales de citas y referencias de las fuentes consultadas.

5. El texto del trabajo final que presento no ha sido publicado ni presentado antes en cualquier medio electrónico o físico.

6. La investigación, los resultados, datos, conclusiones y demás información presentada que atribuyo a mi autoría son veraces.

7. Declaro que mi trabajo final cumple con todas las normas de la Universidad de Piura.

El incumplimiento de lo declarado da lugar a responsabilidad del declarante, en consecuencia; a través del presente documento asumo frente a terceros, la Universidad de Piura y/o la Administración Pública toda responsabilidad que pueda derivarse por el trabajo final presentado. Lo señalado incluye responsabilidad pecuniaria incluido el pago de multas u otros por los daños y perjuicios que se ocasionen.

Fecha: 10/11/2023.

.....

Firma del autor optante[3]

[1] Indicar si es tesis, trabajo de investigación, trabajo académico o trabajo de suficiencia profesional.
[2] Grado de Bachiller, Título profesional, Grado de Maestro o Grado de Doctor.
[3] Idéntica al DNI; no se admite digital, salvo certificado.

UNIVERSIDAD DE PIURA

Declaración Jurada de Originalidad del Trabajo Final

Yo, Franco Augusto Ramirez Cardoza, egresado del Programa Académico de Ingeniería Mecánico-Eléctrica de la Facultad de Ingeniería de la Universidad de Piura, identificado(a) con DNI N° 75142008.

Declaro bajo juramento que:

1. Soy autor del trabajo final titulado:
   "Diseño automático de un tijeral con software Matlab y su verificación con el software de elementos finitos en Solidworks"
   El mismo que presento bajo la modalidad de Tesis[1] para optar el Título profesional[2] de Ingeniero Mecánico Eléctrico.

2. Que el trabajo se realizó en coautoría con los siguientes alumnos de la Universidad de Piura.
   - Diego Angel Ferre Fenco, identificado con DNI N° 72517351
3. La asesoría del trabajo estuvo a cargo de:
   - Dr. Ing. Miguel Buenaventura Castro Sanchez , identificado con DNI N° 02821943
4. El texto de mi trabajo final respeta y no vulnera los derechos de terceros o de ser el caso derechos de los coautores, incluidos los derechos de propiedad intelectual, datos personales, entre otros. En tal sentido, el texto de mi trabajo final no ha sido plagiado total ni parcialmente, para la cual he respetado las normas internacionales de citas y referencias de las fuentes consultadas.

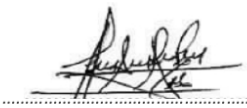
5. El texto del trabajo final que presento no ha sido publicado ni presentado antes en cualquier medio electrónico o físico.

6. La investigación, los resultados, datos, conclusiones y demás información presentada que atribuyo a mi autoría son veraces.

7. Declaro que mi trabajo final cumple con todas las normas de la Universidad de Piura.

El incumplimiento de lo declarado da lugar a responsabilidad del declarante, en consecuencia; a través del presente documento asumo frente a terceros, la Universidad de Piura y/o la Administración Pública toda responsabilidad que pueda derivarse por el trabajo final presentado. Lo señalado incluye responsabilidad pecuniaria incluido el pago de multas u otros por los daños y perjuicios que se ocasionen.

Fecha: 10/11/2023.



Firma del autor optante[3]

[1] Indicar si es tesis, trabajo de investigación, trabajo académico o trabajo de suficiencia profesional.
[2] Grado de Bachiller, Título profesional, Grado de Maestro o Grado de Doctor.
[3] Idéntica al DNI; no se admite digital, salvo certificado.

Dedicatoria

El más sincero agradecimiento a Dios, a mi madre por estar ahí en todo momento, a mi familia y a todas esas grandes personas que siempre me han apoyado en todo momento a lo largo de mi carrera.

Franco Ramirez.

Agradezco a Dios y a la Virgen, por iluminar mi camino a lo largo de estos años, a mi madre Inma por su amor inquebrantable, a mi padre Diego por su constante respaldo y apoyo, a mis abuelos Asunciona, Pablo y Gregorio en la Gloria del Señor, a mi abuela Juana y a toda mi familia y amistades que me acompañaron en este proceso.

Diego Ferre.

Resumen

El presente trabajo de fin de grado tiene como objetivo el diseño y modelado de una estructura metálica tipo tijeral utilizando dos herramientas informáticas: MATLAB y SolidWorks, con la finalidad de poder comparar los resultados entre un lenguaje de programación y un software de diseño CAD 3D.

El primer capítulo de este trabajo, se plantea desarrollar un marco teórico que aborda tanto antecedentes relevantes y conceptos fundamentales para una comprensión más profunda del proyecto. Dentro de este marco, se proporcionará una descripción detallada y una explicación exhaustiva del diseño estructural que se va a implementar. Además, se analizarán en detalle los perfiles estructurales que se utilizarán, se profundizará en las propiedades del acero estructural seleccionado y se explorará de manera minuciosa el método de elementos finitos, todo ello con el propósito de establecer una base sólida para el proyecto.

El segundo capítulo, se realiza un análisis exhaustivo del lenguaje de programación, resaltando la importancia de esta herramienta. Se aborda en detalle el desarrollo de algoritmos, la creación de gráficos y su visualización tanto en 2D como en 3D. También se proporciona, una descripción breve de los elementos de programación, presentados en una tabla con su respectiva explicación. Finalmente, se presenta un diagrama de flujo que ilustra de manera clara el proceso de desarrollo del código para el tijeral.

En el tercer capítulo se enfoca en la implementación del código para el diseño del tijeral utilizando MATLAB. En este capítulo se detalla minuciosamente el código necesario para abordar tres casos distintos, correspondientes a longitudes de base del tijeral de 15, 20 y 27 metros. El objetivo principal de este capítulo es proporcionar una explicación del código del tijeral, con el propósito de obtener resultados óptimos que puedan posteriormente se comparados con los generados por otro software.

En el cuarto capítulo, se procederá a la creación y diseño del tijeral utilizando el software SolidWorks. Paralelamente, se llevará a cabo un análisis estático de este proyecto mediante SolidWorks Simulation, con el objetivo de calcular las fuerzas que afectan a las barras del tijeral. Estos resultados se utilizarán para su posterior comparación con los obtenidos a través de MATLAB.

Tabla de contenido

Lista de tablas

Lista de figuras

Introducción

En la actualidad, la gran mayoría de profesionales en el área de diseño y estructuras, técnicos e ingenieros dedicados a este tema generalmente se toman mucho tiempo en la realización de cálculos y estimación de un elemento estructural idóneo que cumpla con el requerimiento de carga, peso, etc. En esta tesis se busca generar una herramienta más eficaz, capaz de resolver cálculos rápidamente y obtener datos como las cargas, desplazamiento, longitudes, fuerzas y el material estructural necesario para el diseño y construcción del Tijeral.

La razón de la creación de esta tesis es para que tanto los estudiantes y profesionales comprueben que sus cálculos toman sentido al ser aplicados en un caso real, ya que el código MATLAB les mostrará los valores de los cálculos y ellos podrán comprobarlos. De esta manera se pueden afianzar dichos datos obtenidos en MATLAB, en SolidWorks y así poder diseñar y analizar el tijeral de manera segura ahorrando tiempo en los cálculos necesarios. Este proyecto servirá incluso para que los docentes que quieran implementar clases de programas de diseño con un caso real.

Este plan de optimización del proceso de cálculos tiene como objetivo realizar mejoras, obtener cálculos más rápidos, modificaciones e incluso perfeccionamientos de una estructura a partir de simples datos como longitud, largo de divisiones y restricciones que el usuario digitará, así ahorrar tiempo y ganar confiabilidad, para luego optimizar su diseño en un programa CAD 3D.

Capítulo 1
Marco Teórico

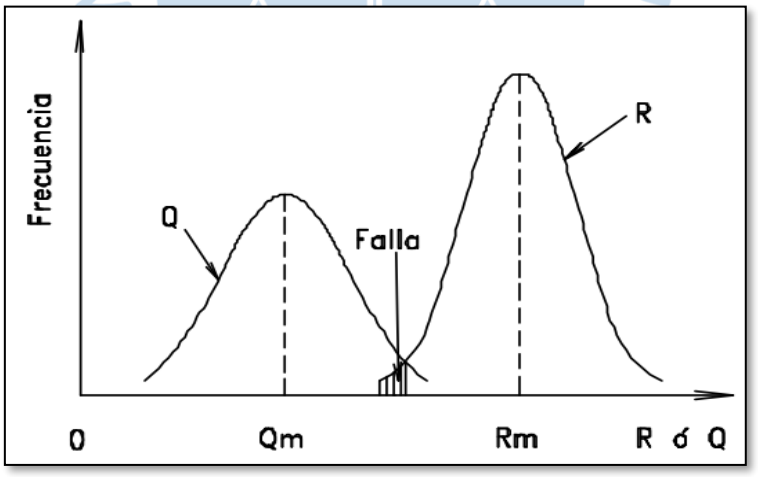
1.1 Método de diseño de estructuras de acero (LRFD)

El método de Diseño de Estructuras de Acero por Factores de Carga y Resistencia, comúnmente abreviado como LRFD (por sus siglas en inglés, Load and Resistance Factor Design), representa un enfoque esencial en el campo de la ingeniería. Su finalidad principal es asegurar tanto la seguridad como la eficiencia en las estructuras de acero. El LRFD ha alcanzado un estatus de reconocimiento a nivel mundial y se ha convertido en el estándar predominante en múltiples normativas de diseño y construcción a nivel global.

El objetivo principal es asegurarse de que la estructura no exceda los límites de seguridad mencionados. Sin embargo, dado que en la práctica es imposible lograr una total ausencia de riesgo, el diseñador debe conformarse con una probabilidad aceptable basadas en métodos estadísticos nombradas como “Métodos de Confiabilidad de momentos de primer orden-segundo orden”. (Zapata, L. 1997)

Figura 1

Conceptos de probabilidades para la determinación del índice de Confiabilidad.



Nota. Obtenido de “Diseño Estructural en Acero” (Zapata, L. 1997)

En la figura 1. se muestra que cuando la resistencia (R) supera la carga (Q), se establece un margen de seguridad. No obstante, también existe la posibilidad contraria, en la que la Resistencia (R) es inferior a la Carga (Q), representada en el área sombreada, lo que constituye una situación de fallo. (Zapata, B. 1997)

1.1.1 Cargas y Factores de Carga (Load Factors)

En el LRFD, las cargas que actúan sobre la estructura se dividen en dos categorías: cargas de servicio y cargas de resistencia. Las cargas de servicio son las que actúan sobre la estructura en condiciones normales de operación, como el peso de las personas, el viento, la nieve, etc. Las cargas de resistencia son las que se utilizan para determinar la capacidad de carga de los componentes estructurales. Se aplican factores de carga a las cargas de servicio para tener en cuenta la variabilidad y la incertidumbre en las condiciones reales. (Goñi, D. y Cáceres, J. 2018)

$$\sum \lambda_i \, x \, Q_i \leq \phi \, x \, R_n \qquad \text{Ecuación 1}$$

Donde:

λ_i: factor de amplificación

Q_i: Carga

ϕ: Factor de reducción

R_n: Resistencia nominal

Es fundamental comprender las diversas fuerzas que actúan en la estructura, por lo que, en las siguientes ecuaciones se mostrarán la combinación de cargas para elementos de acero estructural.

- $P_u = 1.4D$ Ecuación 2
- $P_u = 1.2D + 1.6L + 0.5(S \text{ o } L_r \text{ o } R)$ Ecuación 3
- $P_u = 1.2D + 1.6(L_r \text{ o } S \text{ o } R) + (0.8W \text{ o } 0.5L)$ Ecuación 4
- $P_u = 1.2D + 1.3W + 0.5L + 0.5(L_r \text{ o } S \text{ o } R)$ Ecuación 5
- $P_u = 1.2D + 1.5E + (0.5L \text{ o } 0.2s)$ Ecuación 6
- $P_u = 0.9D - (1.3W \text{ o } 1.5E)$ Ecuación 7

A continuación, se presenta las cargas especificadas.

- ✓ D: Carga muerta
- ✓ L: Carga viva
- ✓ S: Carga de nieve
- ✓ L_r: Carga viva sobre el techo
- ✓ R: Carga de lluvia
- ✓ E: Carga sísmica
- ✓ W: Carga de viento

Los coeficientes de carga aplicados a las cargas muertas son más bajos en comparación con los de las cargas vivas, debido a que los diseñadores pueden calcular con mayor precisión la magnitud de las cargas muertas que de las cargas vivas. Así mismo los valores de carga de servicio que son: D, L, E, L_r, S, W. (Huallpa, R. 2017)

1.1.2 Factores de Resistencia (Resistance Factors)

Los factores de resistencia se aplican a las propiedades de los materiales y a las capacidades de carga de los elementos estructurales para tener en cuenta la variabilidad en la resistencia del material y la incertidumbre en los métodos de análisis. En LRFD, la confiabilidad estructural se evalúa a través de un índice de confiabilidad, que se calcula mediante un análisis estadístico de las cargas y las capacidades de resistencia. Los factores de carga y resistencia se determinan de manera que la confiabilidad de una estructura alcance el nivel deseado, siguiendo las pautas de la normativa. (Faragó, R., Sousa, M. y Riqueira. A 2022)

Este índice de confiabilidad guarda una relación directa con los factores de carga y resistencia utilizado en el diseño, por lo cual en la Tabla 1 se presenta los valores de los factores de resistencia para conectores o miembros.

Tabla 1

Valores de los factores de resistencia.

Valor	Miembro o Conector
0.90	Área bruta en tracción
0.75	Área neta de conexión en tracción
0.90	Miembros en flexión
0.85	Miembros en compresión axial
0.75	Pernos en tracción

Nota. Obtenido de “Diseño Estructural en Acero” (Zapata, L. 1997)

1.2 Acero Estructural

"Para su bien o para su mal el material acero es uno de los materiales que más ha influido en la vida del hombre; es agente de adelanto y civilización, de destrucción y miseria, de bienestar y libertad, de poder y opresión. El arado y la espada, que caracterizan a la humanidad, son de acero". (Zapata, B. 1997)

Dentro de la amplia variedad de aceros disponibles, los más relevantes en el ámbito de la construcción son los aceros estructurales. Estos aceros están diseñados específicamente para soportar cargas y deben ser producidos de acuerdo con estrictas normativas de fabricación para garantizar su idoneidad y resistencia. (Cruz, E y Hancco D. 2023)

Tabla 2

Propiedades mecánicas del acero estructural.

Propiedades mecánicas	
Peso específico	$7850\ \text{kgf/m}^3$
Módulo de elasticidad longitudinal (Módulo de Young)	$2.1\ x\ 10^6\ \text{kgf/cm}^2$
Módulo de elasticidad transversal (Módulo de corte)	$G = \frac{E}{2(1+\nu)}$
Coeficiente de Poisson	$\nu = 0.3$ (rango elástico) $\nu = 0.5$ (rango plástico)
Coeficiente de dilatación térmica	$11.7\text{x}\ 10^{-6}/°\text{C}$

Nota. Obtenido de "Diseño estructural en acero para el mejoramiento de los servicios de educación secundaria en la I.E. Champagnat en el distrito de Tacna, provincia de Tacna– Tacna- Perú" (Ramos, R. 2022).

Tener en cuenta: Es esencial resaltar que el coeficiente de Poisson puede experimentar notables variaciones según el tipo de material y el nivel de carga involucrado. En el rango elástico, suele situarse alrededor de $\nu = 0.3$, pero en el rango plástico, es común encontrar valores cercanos a $\nu = 0.5$. Sin embargo, es importante destacar que este coeficiente es altamente susceptible a cambios, incluso entre diferentes clases de aceros estructurales. Por lo tanto, se recomienda realizar pruebas específicas o consultar las especificaciones del material en uso para obtener valores precisos del coeficiente de Poisson.

Para este proyecto, se utilizarán las normas ASTM como referencia para la caracterización de los diversos Grados de Aceros disponibles en el mercado internacional, en particular para los perfiles.

Aplicaciones de los aceros estructurales según las normas ASTM.

- A36: Para aplicaciones generales en construcciones, particularmente en edificios comerciales, residenciales e industriales, tanto en la estructura principal como en componentes como vigas, columnas, placas base y conexiones.
- A242: Para estructuras de puentes que requieran ser conectadas mediante pernos o soldaduras, y que posean una alta resistencia a la corrosión.
- A572: Para perfiles estructurales, láminas y barras utilizadas en construcciones que requieran ser ensambladas mediante pernos o soldaduras, y para puentes, se emplearan soldaduras únicamente en los Grados 42 y 50.

Dado que el acero ASTM A36 presenta una alta resistencia a la tracción de aproximadamente 400 MPa y una resistencia al rendimiento de alrededor de 250 MPa, además de ser reconocido por su ductilidad y tenacidad, lo convierte en una elección altamente apropiada para su aplicación en la construcción y fabricación. Por lo tanto, se ha optado por utilizar esta norma ASTM A36. En el territorio peruano, las empresas Sider-Perú en su instalación de Chimbote y Aceros Arequipa SA en su complejo de Pisco, son los principales fabricantes de ciertos productos, como la palanquilla. Esta palanquilla se emplea en el proceso de laminación de ángulos y varillas lisas, y su composición es bastante similar a la del Acero ASTM A36. (Zapata, L. 1997)

1.3 Diseños Estructurales

1.3.1 Miembros en tracción

Los elementos de las estructuras que experimentan esfuerzos internos que impiden la separación de sus extremos bajo la influencia de una carga axial son conocidos como “Miembros en Tracción Axial”. Estos componentes son esenciales para asegurar la integridad y la resistencia de las estructuras en las que se utilizan.

Para el caso de los Miembros en Tracción, se identifican dos estados límites fundamentales: Fluencia y Fractura.

Tabla 3

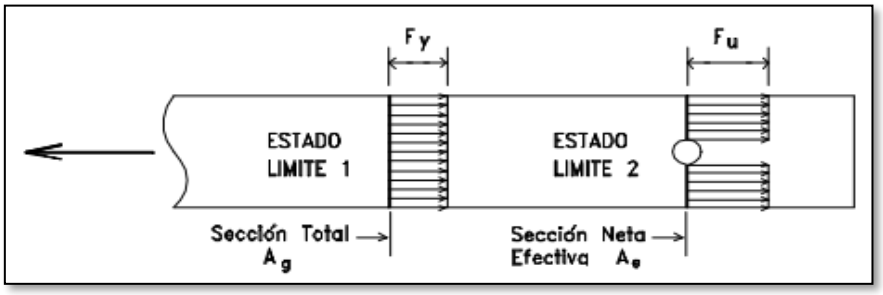
Estados límites de miembros en tracción.

Caso	Estados límites	Factor de resistencia	Coeficiente de seguridad
Miembros en tracción	Fluencia en el área bruta	0.90	1.67
	Rotura en el área neta	0.75	2.00

Nota. Obtenido de comparativo técnico-económico de una nave industrial con un sistema de tijerales y de pórticos (Goñi, D., y Cáceres, J. 2018)

Figura 2

Estados límites de miembros en tracción



Nota. Obtenido de "Diseño Estructural en Acero" (Zapata, L. 1997)

A continuación, se detalla las descripciones de cada uno de los límites de estados mencionados anteriormente.

- ✓ Fluencia en el área total

$$P_{nf} = \phi * P_n = \phi * F_y * A_g \qquad \text{Ecuación 8}$$

Donde:

$\phi = 0.90$

P_n: Resistencia nominal

F_y: Esfuerzo de fluencia de acero

A_g: Área total

✓ Rotura en el área neta

$$P_{nr} = \phi * P_n = \phi * F_u * A_e \qquad \text{Ecuación 9}$$

Donde:

$\phi = 0.75$

P_n: Resistencia nominal

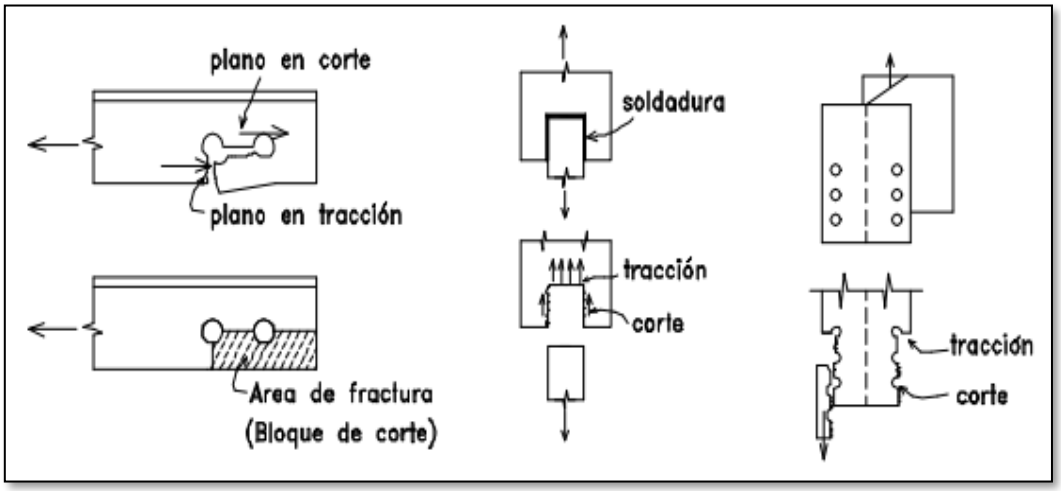
F_u: Esfuerzo de rotura en la sección neta

A_e: Área neta efectiva

1.3.1.1 Bloque de corte. Este tipo de fallo se ha observado previamente en vigas copadas, y ahora se ha vuelto evidente que este límite de estado también influye en ocasiones en el comportamiento de las extremidades conectadas de los miembros que están sujetos a tracción. En las uniones de las extremidades, no siempre se determinará el camino o menor resistencia por la sección transversal nomina (An o Ae). En cambio, existe un camino de fallo que engloba dos planos: tracción en uno y corte en el plano perpendicular al otro. En algunos casos, esta situación puede resultar más crítica, como se ilustra en la Figura 3.

Figura 3

Bloque de corte



Nota. Obtenido de "Diseño Estructural en Acero" (Zapata, L. 1997)

La falla que implica tracción en un plano y corte simultáneo en un plano perpendicular recibe el nombre de "bloque de corte". (Zapata, L. 1997). En consecuencia, existen dos modalidades posibles de fallo:

- Fractura de tracción (Fu) con fluencia de corte (0.6 Fy)
- Fractura de corte (0.6 Fu) con fluencia de tracción (Fy)

1.3.1.2 Relación de Esbeltez de miembros en tracción: L/r. A pesar de que los miembros en tracción no son propensos al pandeo, las especificaciones AISC-LRFD establecen que, en los miembros traccionados, el valor de L/r (longitud efectiva dividida por el radio de giro) no debe superar 300, siendo preferible esta limitación (a excepción de las varillas, que no están sujetas a esta restricción). La razón detrás de la recomendación de este límite es facilitar la fabricación y el manejo durante el montaje, así como prevenir la formación de ondas por el calor en el caso de uniones soldadas entre los perfiles. Además, se requiere una relación L/r aún menor para miembros que estarán expuestos a la acción del viento por su propio peso, lo que podría inducir flexión, o para aquellos que están sujetos a vibraciones causadas por maquinarias. (Zapata, L. 1997).

1.3.2 Miembros en compresión

En el caso de los miembros en tracción, se ha observado que estos componentes pueden soportar cargas sin experimentar problemas de pandeo interno, y la preocupación solo secundaria, centrándose ocasionalmente en la longitud o la configuración de su sección transversal.

Por otro lado, en el estudio de los miembros en compresión axial, la resistencia a las cargas aplicadas depende, entre otros factores, de la longitud efectiva del miembro y de la geometría de su sección transversal. La longitud efectiva, a su vez, está influenciada por el tipo de conexiones presentes y el grado de desplazamiento relativo entre sus nodos.

Es importante destacar que los miembros en compresión están sujetos al fenómeno del pandeo. A medida que la carga axial de compresión aplicada aumenta, se llega a un punto en el que se produce un tipo de inestabilidad conocido como “carga de pandeo”, en la cual el miembro pierde su capacidad de soportar la carga de manera eficiente.

Tabla 4

Estados límites de miembros en compresión

Caso	Estados Límites	Factor de resistencia	Coeficiente de seguridad
Miembros en compresión	Pandeo flexional	0.9	1.67
	Pandeo flexo-torsional o torsional	0.85	1.67
	Pandeo local	0.9	1.67

Nota. Obtenido de comparativo técnico-económico de una nave industrial con un sistema de tijerales y de pórticos (Goñi, D., y Cáceres, J. 2018)

Los profesionales del diseño suelen utilizar tablas que ofrecen los esfuerzos críticos para cargas de compresión axial basadas en la relación Kl/r. Con las nuevas fórmulas del AISC-LRFD, esto simplifica gracias a la aceleración directa existente entre ambos valores. En el apéndice se incluye una tabla específica para acero con una resistencia nominal (Fy) de 2530 kg/cm2, con el propósito de facilitar la aplicación de las fórmulas para el cálculo del esfuerzo crítico. (Goñi, D. y Cáceres, J. 2018)

Para llevar a cabo este proceso, es esencial considerar la expresión que determina el límite máximo de esbeltez permitida.

$$\frac{KL}{r} \leq 200$$ Ecuación 10

Donde:

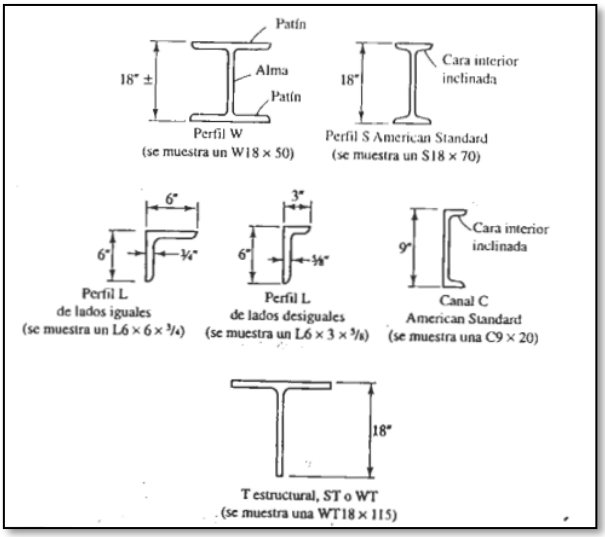
- K: Factor de longitud
- L: Longitud sin arriostrar
- r: Radio de giro

1.4 Perfiles estructurales

El proceso de diseño de estructuras conlleva la elección cuidadosa de perfiles estructurales apropiados, con el propósito de garantizar la capacidad de soportar las cargas previstas y satisfacer los estándares de resistencia y estabilidad requeridos para cualquier tipo de edificio o estructura. Estos perfiles estructurales desempeñan un papel esencial en le edificación de una diversidad de estructuras, que abarcan desde edificios hasta puestes y torres. “La selección de un perfil comercial será casi siempre la opción más económica, incluso si ello implica usar un poco más de material. (Segui, W. 2000)

Figura 4

Secciones de los perfiles estructurales.

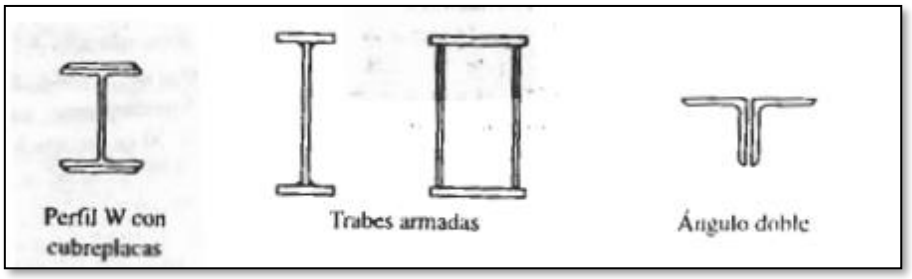


Nota. Obtenido de Diseño de estructuras de acero con LRFD (Segui, W. 2000)

En la figura 4 se presentan los perfiles más utilizados, aunque existen otros perfiles disponibles. En la mayoría de los casos, uno de estos perfiles estándar será suficiente para cumplir con los requisitos del diseño. Sin embargo, si los requisitos son particularmente exigentes, podría ser necesario recurrir a una sección compuesta, como se ilustra en la figura 5. Esto cuando ninguno de los perfiles estándar es lo suficientemente grande, es decir, cuando la sección transversal carece de la suficiente área o momento de inercia. (Segui, W. 2000)

Figura 5

*Secciones compuestas de perfiles estructurales***.**



Nota. Obtenido de Diseño de estructuras de acero con LRFD (Segui, W. 2000)

1.5 Método de los elementos finitos

La mayoría de las estructuras en ingeniería poseen una naturaleza continua y tridimensional intrínseca. Sin embargo, en ciertos casos, es posible describir su comportamiento de manera adecuada mediante modelos matemáticos que son unidimensional o bidimensional.

Cuando se abordar un problema estructural, generalmente se busca obtener información sobre los desplazamientos, deformaciones y tensiones en cualquier punto del dominio en cuestión. Para lograrlo, de acuerdo con el Método de Elementos Finitos (MEF), se fragmenta el dominio en subdominios, conocidos como elementos finitos, y se crea una malla que consiste en una disposición discreta de estos elementos finitos, junto con sus bordes y nodos correspondientes (Morató, J. 2015)

El punto de partida fundamental para cualquier análisis estructural es el modelo geométrico. En el contexto de herramientas basadas en métodos numéricos de aproximación, como el Método de Elementos Finitos (MEF), se lleva a cabo un proceso de discretización o mallado del modelo. Este proceso implica dividir la geometría en elementos relativamente pequeños y de forma simple, conocidos como elementos finitos. Estos elementos finitos reciben su nombre para resaltar que no son infinitamente pequeños, sino que tienen un tamaño razonable en comparación con la escala total del modelo. Cuando se emplean elementos finitos en el análisis, el solucionador del MEF se aproxima a la solución deseada, como las deformaciones o las tensiones, para todo el modelo mediante el conjunto de soluciones para cada elemento individual. (González, O., González, C., López, A. 2019)

Desde la perspectiva de un software de análisis de elementos finitos, el proceso implica tres etapas principales:

1. Preprocesamiento: En esta etapa, se definen aspectos como el tipo de análisis a realizar (por ejemplo, estático, térmico, de frecuencia), las propiedades del material, cargas y las restricciones. Además, se lleva a cabo la división del modelo en elementos finitos.
2. Solución Computacional: En esta fase, se realiza el cálculo numérico para obtener los resultados deseados, como desplazamientos, tensiones, etc.
3. Postprocesamiento: Una vez obtenidos los resultados, se procede a analizar y visualizar los mismos para extraer información relevante.

Desde la perspectiva de la metodología del Análisis de Elementos Finitos, los pasos se resumen en los siguientes:

1. Formulación del modelo matemático.
2. Creación del modelo de elementos finitos.
3. Resolución del modelo de elementos finitos.
4. Análisis de los resultados.

Capítulo 2
Reseña de Lenguaje de programación en MATLAB

2.1 Introducción del software MATLAB

El lenguaje de programación MATLAB, cuyo nombre es un acrónimo de "MATrix LABoratory", es una herramienta ampliamente reconocida y utilizada en el ámbito científico y de la ingeniería. Desarrollado por MathWorks, MATLAB se ha convertido en una pieza fundamental en investigaciones y análisis numéricos en diversas disciplinas académicas. La flexibilidad y eficiencia de MATLAB en el manejo de datos numéricos y su capacidad para realizar análisis y visualizaciones precisas, lo convierten en un recurso invaluable para investigadores y profesionales. (MathWorks, 2012)

El objetivo principal de esta tesis es la exploración exhaustiva del lenguaje de programación MATLAB y su aplicación en la solución de problemas complejos en ciencia e ingeniería. Durante el desarrollo de este este trabajo, se analizarán a detalle las características esenciales de MATLAB, su sintaxis y su capacidad para abordar una variedad de desafíos computacionales.

Además, se investigarán ejemplos específicos de aplicaciones de MATLAB en áreas como análisis de datos, modelado matemático y simulación. Esta tesis también analizará la importancia de MATLAB en el contexto de la investigación científica y su contribución a la eficiencia y precisión en el análisis de datos y la toma de decisiones basada en evidencia.

Esta tesis se centra en explorar el potencial de MATLAB en el diseño estructural de tijerales, destacando su papel en la simulación, optimización y análisis de estos elementos cruciales en la ingeniería y arquitectura. A lo largo de este trabajo, se examinarán ejemplos concretos de aplicaciones de MATLAB en el diseño de tijerales, mostrando como esta herramienta mejora la eficacia y exactitud en la construcción de estructuras que cumplen con los estándares de seguridad y funcionalidad requerido.

2.2 Programación y desarrollo de algoritmos

MATLAB proporciona un entorno de desarrollo en el que se puede programar y analizar algoritmo y aplicaciones de manera eficiente. Su lenguaje de programación ofrece soporte integrado para operaciones con matrices y vectores, aspecto critico en la resolución de problemas en campos como la ingeniería y ciencia. Esto agiliza el proceso de desarrollo y ejecución de programas.

Una de las ventajas sobresalientes de MATLAB radica en su capacidad para acelerar la programación y el diseño de algoritmos en comparación con lenguajes tradicionales. Al eliminar tareas de bajo nivel, como la declaración de variables y la gestión de memoria, los programadores pueden escribir programas y desarrollar algoritmos más rápidamente.

Además, MATLAB y sus productos complementarios ofrecen una amplia variedad de algoritmos incorporados en áreas como el procesamiento de señales y comunicaciones, procesamiento de imágenes y videos, sistemas de control, diseño de estructuras metálicas y otros dominios. Combinando estos algoritmos con los propios del usuario, es posible desarrollar aplicaciones y programas complejos. (Sanchez, R. 2014)

2.2.1 Entorno de Desarrollo Integrado

MATLAB proporciona diversas utilidades destinadas a la eficiente creación de algoritmos, las cuales abarcan:

- Ventana de comandos (Command Window): Ventana interactiva en la que los usuarios pueden ingresar comandos de MATLAB y realizar cálculos en tiempo real. Los resultados de las operaciones se muestran de inmediato en esta ventana, lo que facilita la exploración y ejecución de comandos de manera rápida y sencilla.
- Editor de código (Editor): Herramienta que permite crear, modificar y gestionar scripts y funciones de MATLAB de manera eficiente. Proporciona características como el resaltado de sintaxis, la corrección automática y la organización estructura del código, lo que facilita la escritura y mantenimiento de programas.
- Espacio de trabajo (Workspace): Parte esencial del entorno de desarrollo que muestra las variables que están actualmente en uso durante la sesión de MATLAB. Proporciona información sobre el nombre de la variable, su tipo de dato y su contenido, lo que facilita el seguimiento y la inspección de datos en tiempo real.
- Carpeta actual (Current Folder): Proporciona acceso rápido a los archivos y carpetas en el sistema de archivos de la computadora en uso. Esto es esencial para cargar y guardar archivos de datos, scripts y funciones, lo que simplifica la gestión de archivos en el contexto de la sesión de MATLAB.
- Historial de Comandos (Command History): Registra los comandos previamente ejecutados en la sesión actual. Esto permite a los usuarios recuperar y reutilizar comandos anteriores, lo que es útil para realizar tareas repetitivas o para recordar acciones realizadas anteriormente.

2.2.2 Gráficos y visualización

Una de las características sobresalientes de MATLAB es su capacidad avanzada para crear, personalizar y visualizar gráficos e imágenes de manera efectiva. Estas capacidades son fundamentales en la investigación y el análisis de datos en diversas disciplinas científicas y técnicas. A continuación, se detallan las principales características de gráficos y visualización en MATLAB:

- Creación de gráficos: MATLAB ofrece una amplia variedad de funciones y comandos para crear gráficos de calidad profesional. Estas funciones incluyen “plot()” para trazar

gráficos de líneas, “scatter()” para diagramas de dispersión, “bar()” para gráficos de barras y muchas otras. Los usuarios pueden representar datos de manera efectiva utilizando diferentes tipos de gráficos según sus necesidades.

- Personalización de gráficos: Personalización exhaustiva de gráficos. Los usuarios pueden modificar colores, estilos de línea, tamaños de fuente, etiqueta de ejes y más para adaptar los gráficos a sus requisitos específicos. Esta capacidad es esencial para la presentación visual de resultados de investigación.
- Visualización 2D y 3D: MATLAB admite tanto gráficos en 2D como en 3D, lo que permite a los investigadores visualizar datos en múltiples dimensiones. Los gráficos en 3D son cruciales para representar datos tridimensionales, como superficies y volúmenes.
- Exportación de gráficos: MATLAB permite funciones para la creación de gráficos interactivos, lo que permite a los usuarios explorar datos y realizar análisis directamente en los gráficos. Esto es especialmente útil en la investigación exploratoria de datos.

2.3 Descripción básica de los elementos de programación

En el entorno de programación MATLAB, los elementos de control son herramientas esenciales que permiten a los desarrolladores y científicos controlar el flujo de ejecución de sus programas. Estos elementos son esenciales para tomar decisiones, realizar operaciones repetitivas y estructurar lógica de un programa de manera eficiente. Cada uno de los elementos de la Tabla 5 desempeña un papel importante en la programación y el análisis de datos en MATLAB, permitiendo a los usuarios diseñar algoritmos robustos y solucionar problemas de manera efectiva.

Tabla 5

Estados límites de miembros en compresión

Elemento	Descripción
clc	Limpia la ventana de comandos, eliminando resultados y comandos anteriores para mantenerla ordenada.
clear all	Borra todas las variables almacenadas en la memoria de MATLAB, lo que puede ser beneficioso para liberar memoria y evitar confusiones debido a la presencia de variables no deseadas.
syms	Declara símbolos simbólicos, lo que permite realizar cálculos algebraicos y simbólicos en lugar de numéricos.

for	Herramienta que se emplea para generar repeticiones de comandos, según un número predefinido de iteraciones
if	Se utiliza para tomar decisiones en un programa. Permite ejecutar un bloque de código si una condición dada es verdadera.
elseif	Evalúa condiciones adicionales si la primera condición es falsa.
end	Marca el final de bucles, bloques de código y asignaciones de índices.
fprintf	Función para imprimir texto formatear y escribir datos en un archivo o en la ventana de comando de MATLAB.
find	Localiza elementos que cumplen con una condición en una matriz o vector, proporcionando sus posiciones.
switch	Estructura de control utilizada para tomar decisiones múltiples basadas en el valor de una expresión. Permite ejecutar diferentes bloques de código según el valor de la expresión evaluada.
figure	Crea una ventana gráfica, que permite mostrar gráficos y visualizaciones en una ventana separada.
plot	Se emplea para generar gráficos, trazar datos y visualizar información en una ventana gráfica.
hold on	Comando que permite superponer múltiples gráficos en la misma ventana gráfica en MATLAB, en lugar de reemplazar el gráfico existente por uno nuevo.
size	Determina las dimensiones (números de filas y columnas) de una matriz o vector, para comprender su estructura de los datos.
zeros	Se utiliza para crear una matriz de ceros con dimensiones especificadas.
ones	Se emplea para generar una matriz de unos con dimensiones especificadas.
length	Permite determinar la longitud de un vector o la dimensión más larga de una matriz.

text	Agrega texto a gráficos.
line	Dibuja líneas en gráficos.
num2str	Convierte números a texto.
fopen	Abre archivos y estable una conexión entre programa y el archivo.
fclose	Cierra archivos previamente abiertos con fopen.
quiver	Crea gráficos de vectores que representan campos vectoriales.
disp	Se emplea para mostrar texto o contenido en la ventana de comandos.
input	Obtener datos ingresados por el usuario a través de la ventana de comandos.
strcat	Se utiliza para concatenar(unir) múltiples cadenas de texto en una sola cadena.
dlmerad	Lee datos numéricos desde archivos de texto con formato delimitado, como archivos CSV.
readtable	Lee datos tabulares desde archivos, como hojas de cálculo de Excel o archivos CSV.
table2array	Convierte una tabla de datos en una matriz.
table2cell	Convierte una tabla de datos en una estructura de celda.
max	Encontrar el valor máximo en un conjunto de datos.
min	Encontrar el valor mínimo en un conjunto de datos.
abs	Calcula el valor absoluto de los elementos de una matriz.
find	Busca y proporciona las posiciones de elementos que cumplen con una condición específica en una matriz o vector.
floor	Redondea hacia abajo un número decimal a su valor entero más cercano inferior.

2.4 Diagrama de flujo

Figura 6

Inicio del Código Tijeral.m

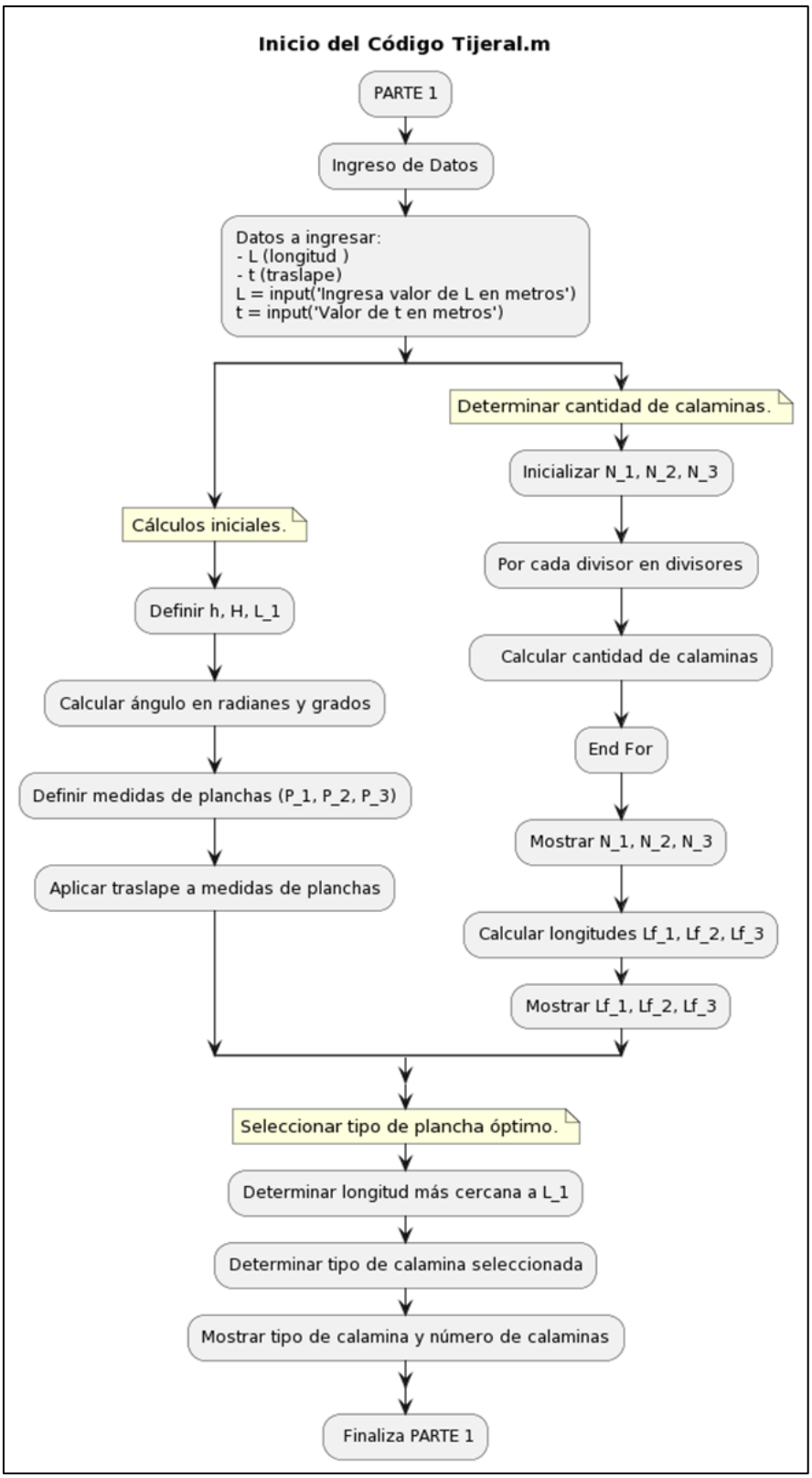


Figura 7

Segunda Parte del Código Tijeral.m

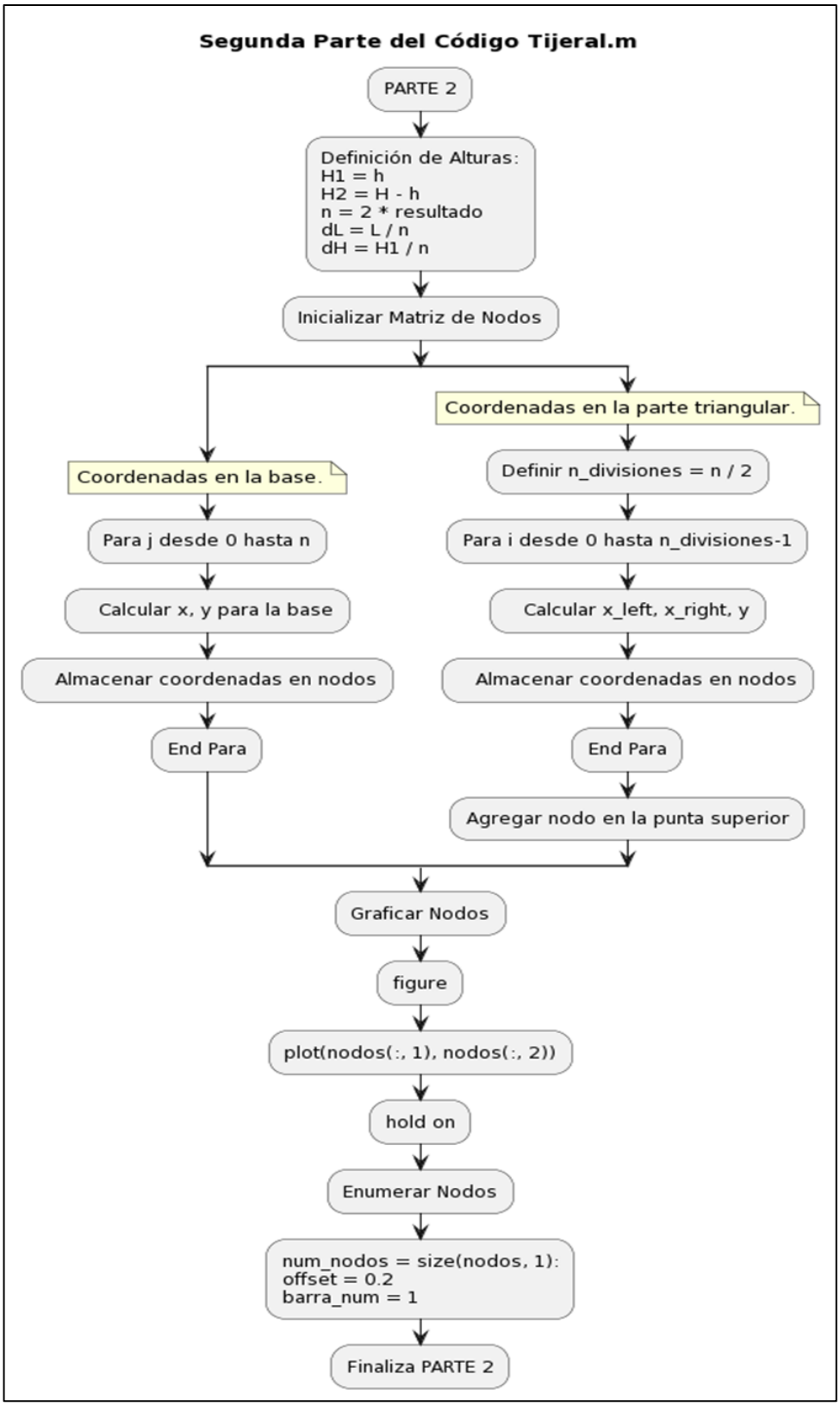


Figura 8

Tercera Parte del Código Tijeral.m

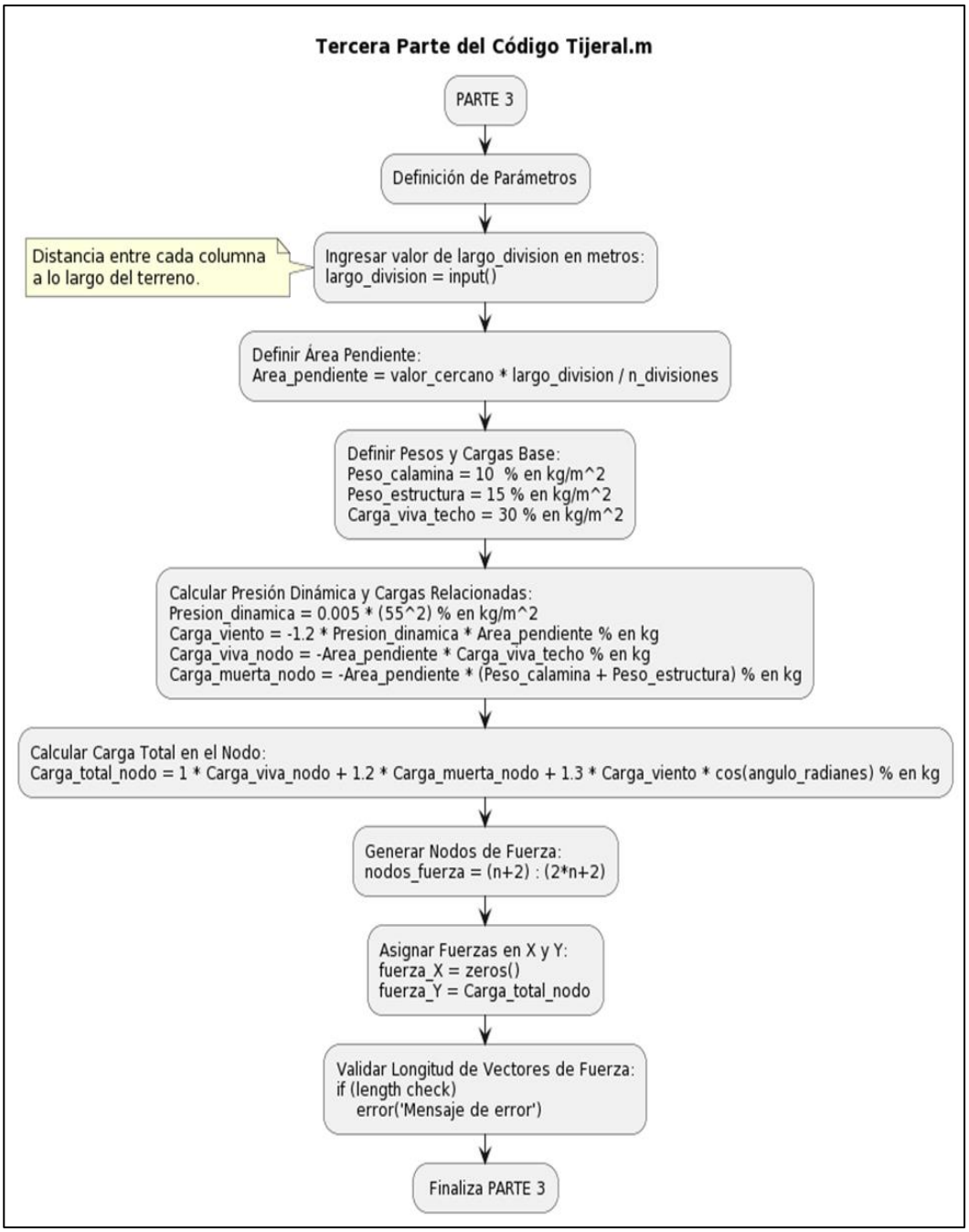


Figura 9

Cuarta Parte del Código Tijeral.m

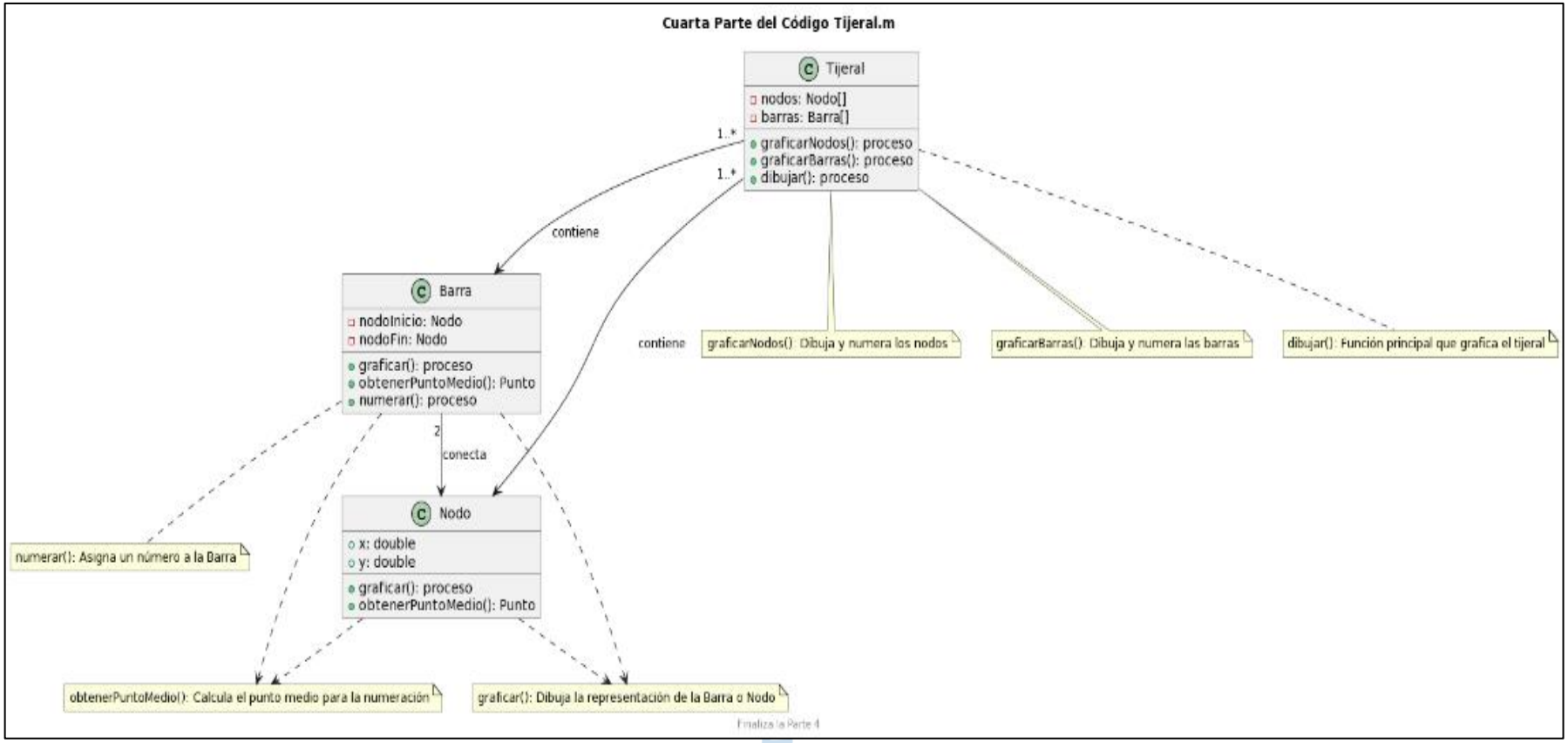


Figura 10

Quinta Parte del Código Tijeral.m

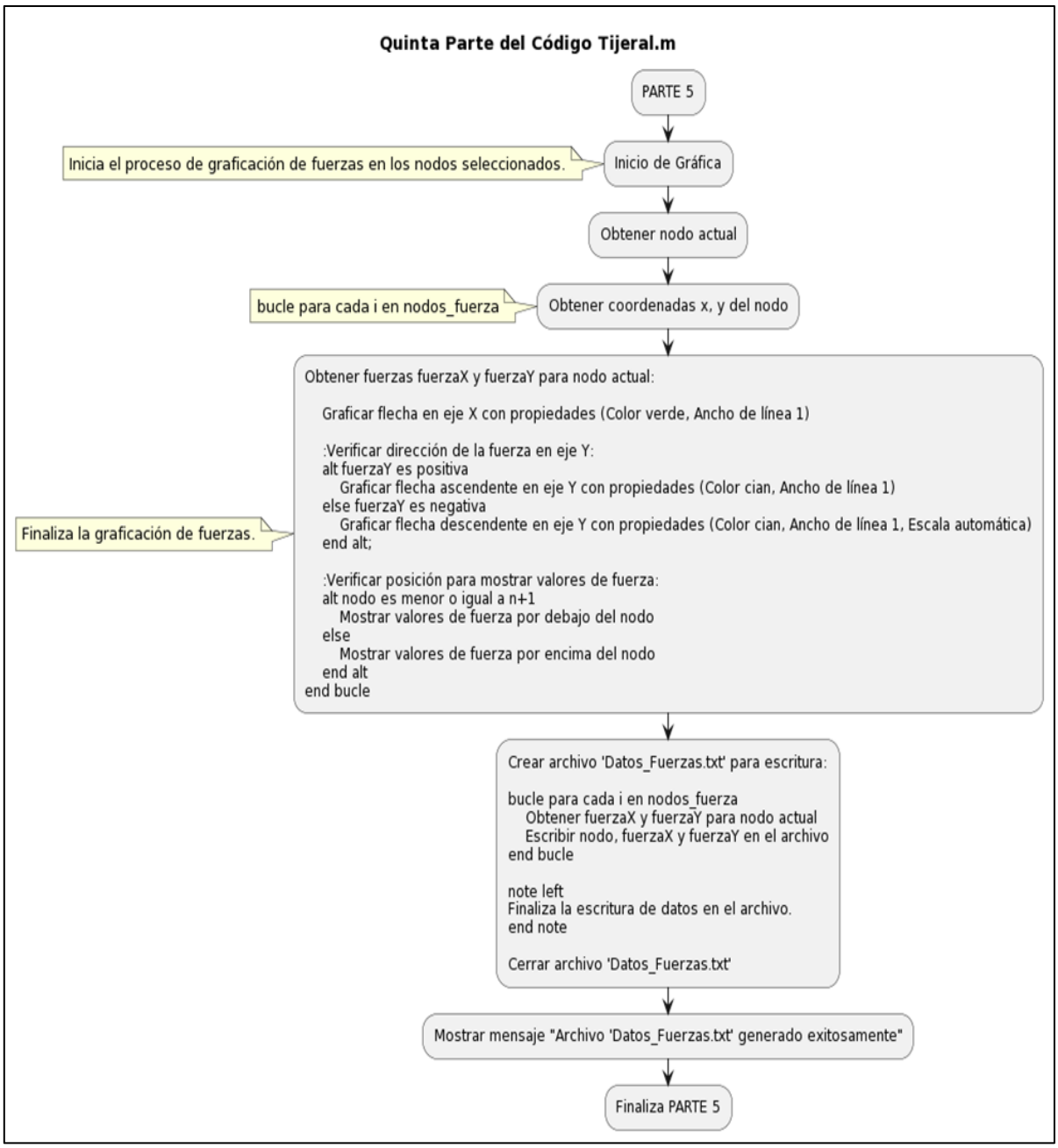


Figura 11

Sexta Parte del Código Tijeral.m

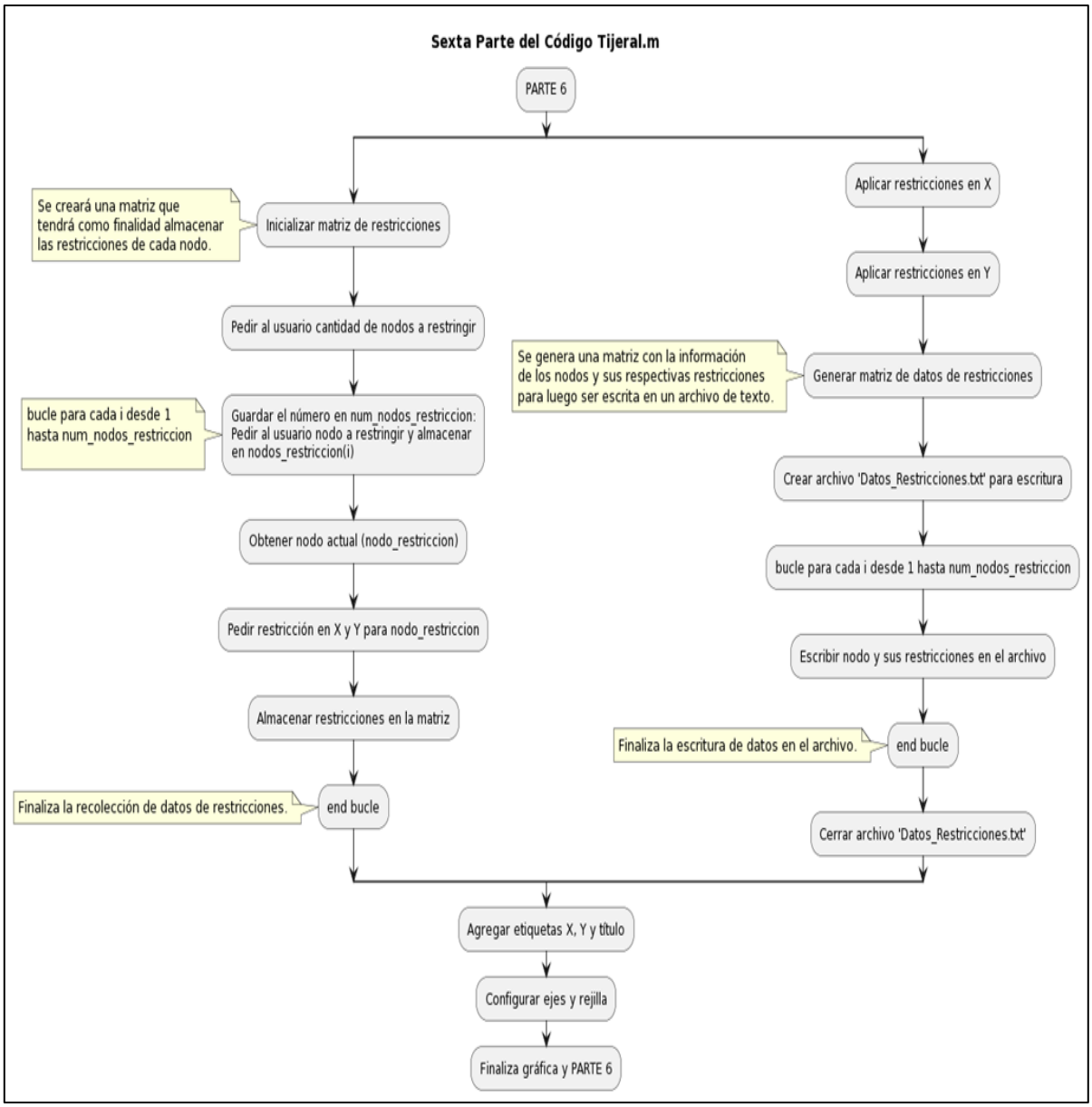


Figura 12

Séptima Parte del Código Tijeral.m

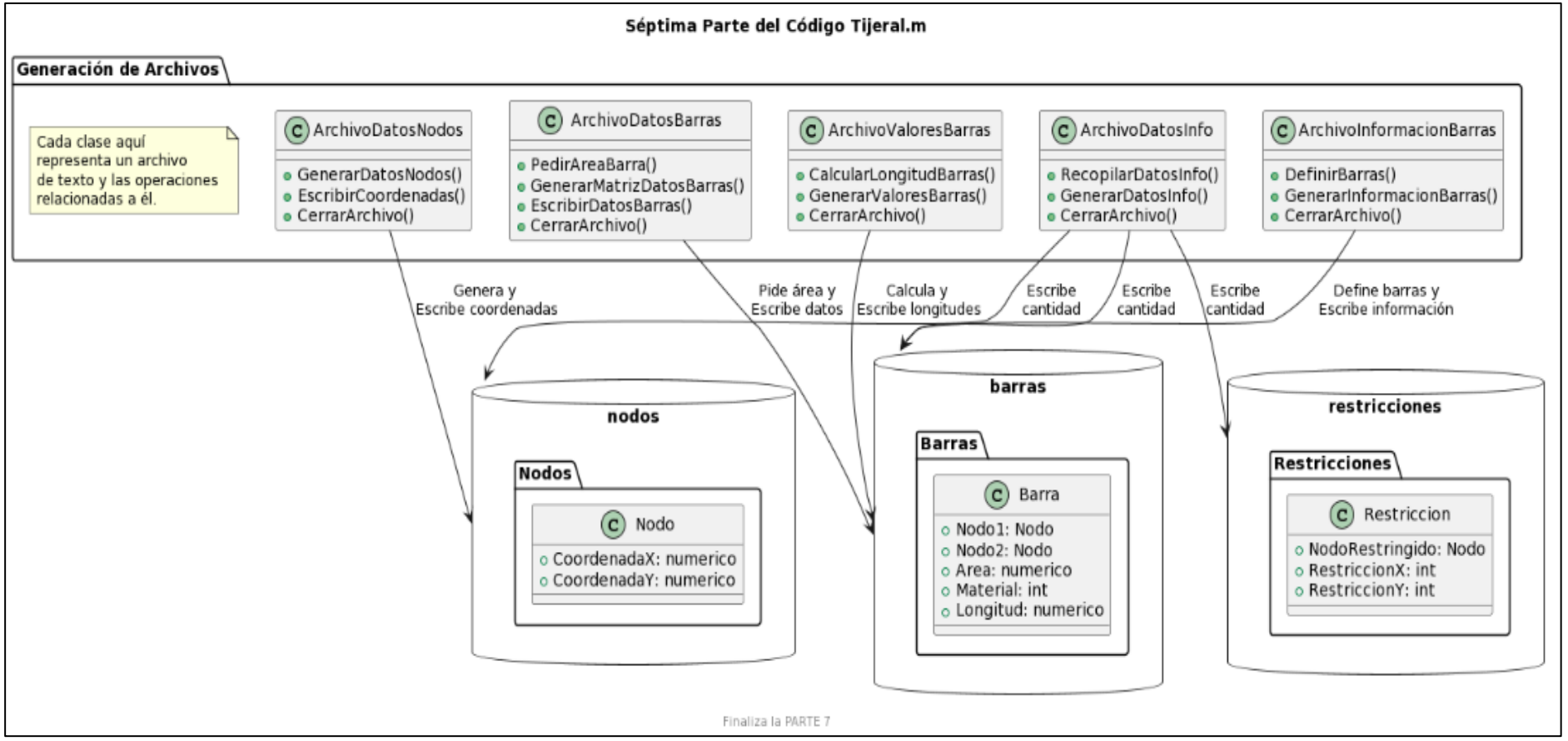


Figura 13

Primera Parte del Código Truss_Tijeral.m

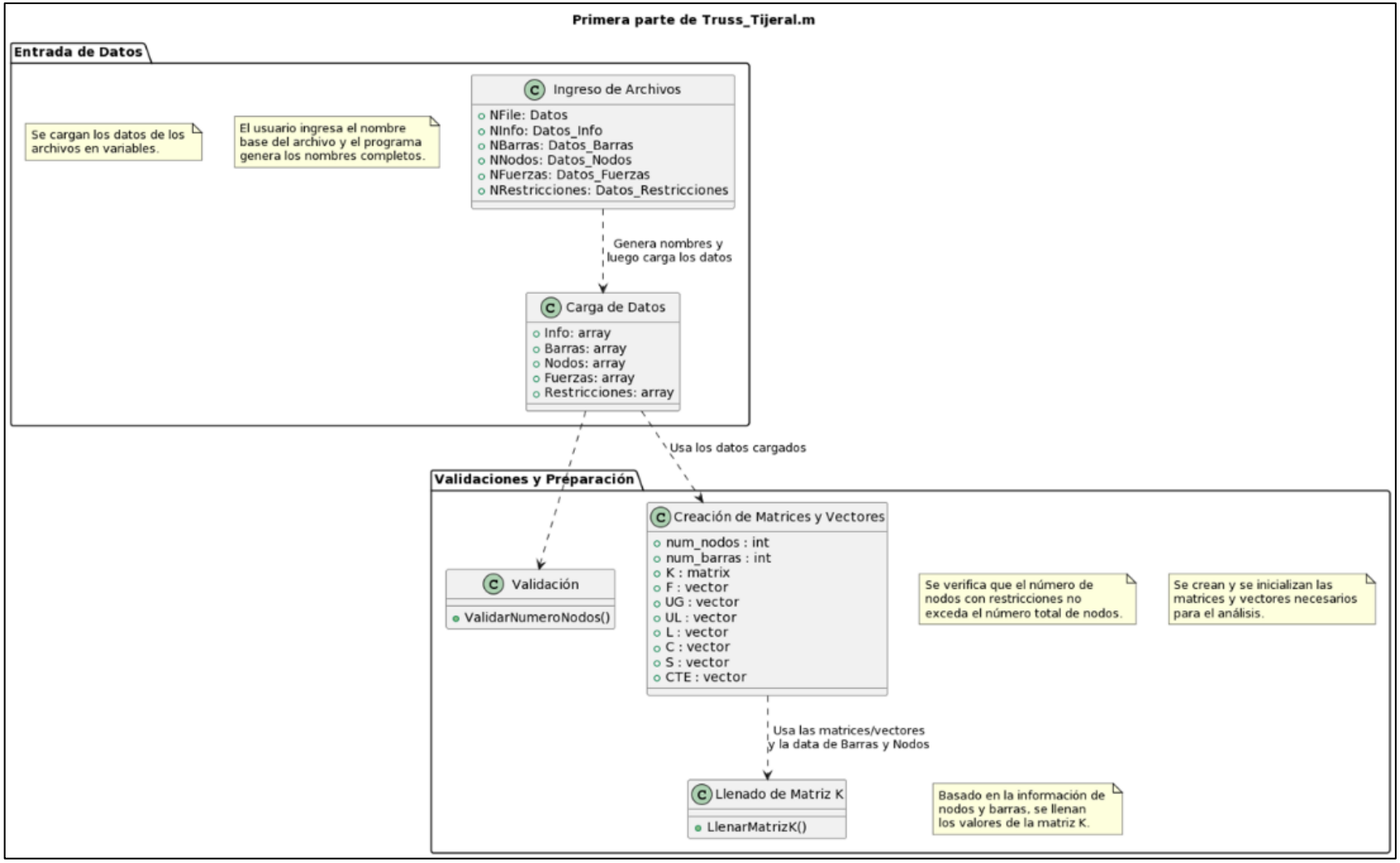


Figura 14

Segunda Parte del Código Truss_Tijeral.m

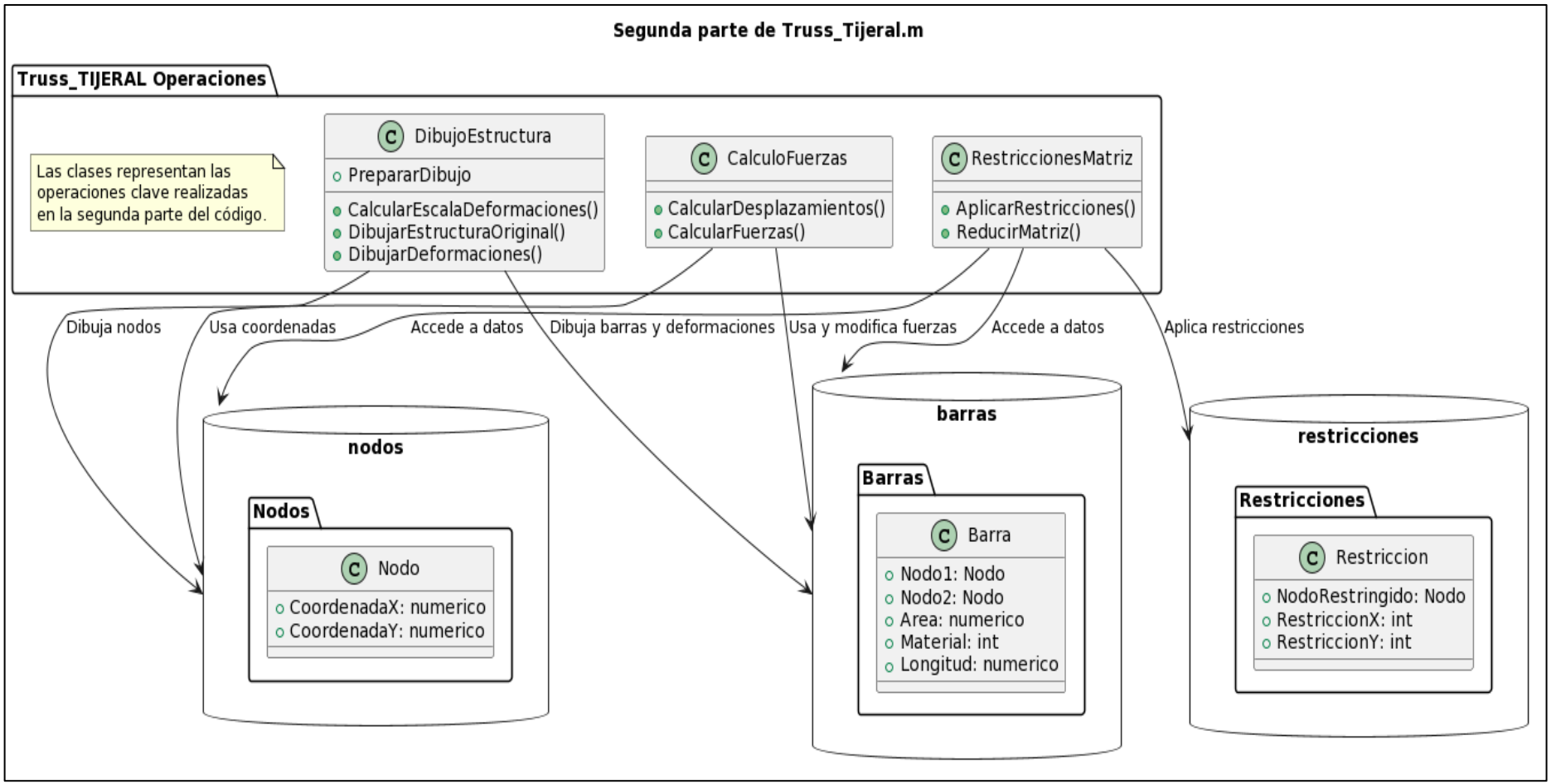


Figura 15

Tercera Parte del Código Truss_Tijeral.m

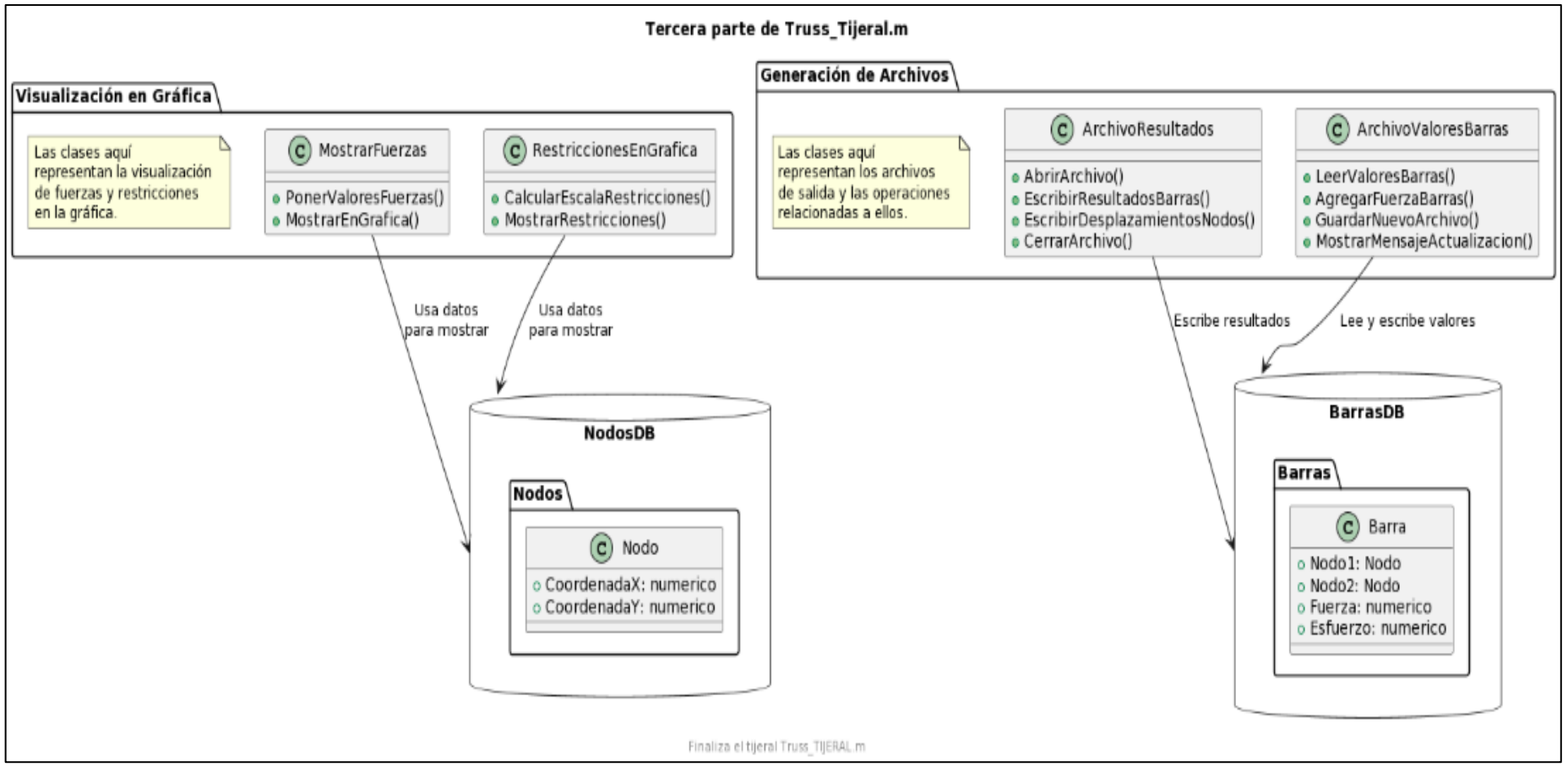


Figura 16

Primera Parte del Código SeleccionPerfil.m

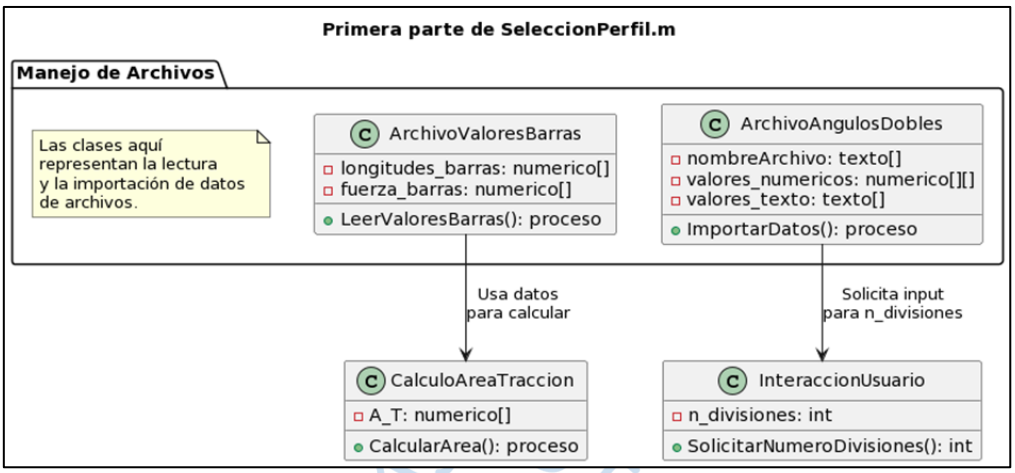


Figura 17

Segunda Parte del Código SeleccionPerfil.m

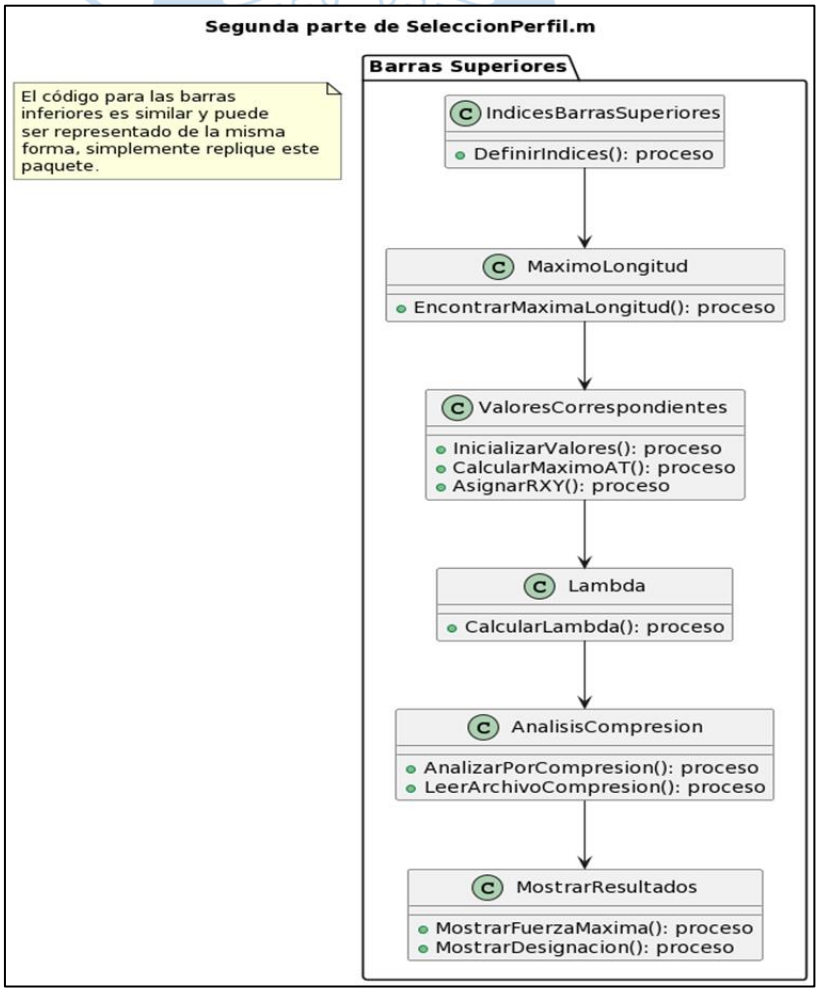


Figura 18

Tercera Parte del Código SeleccionPerfil.m

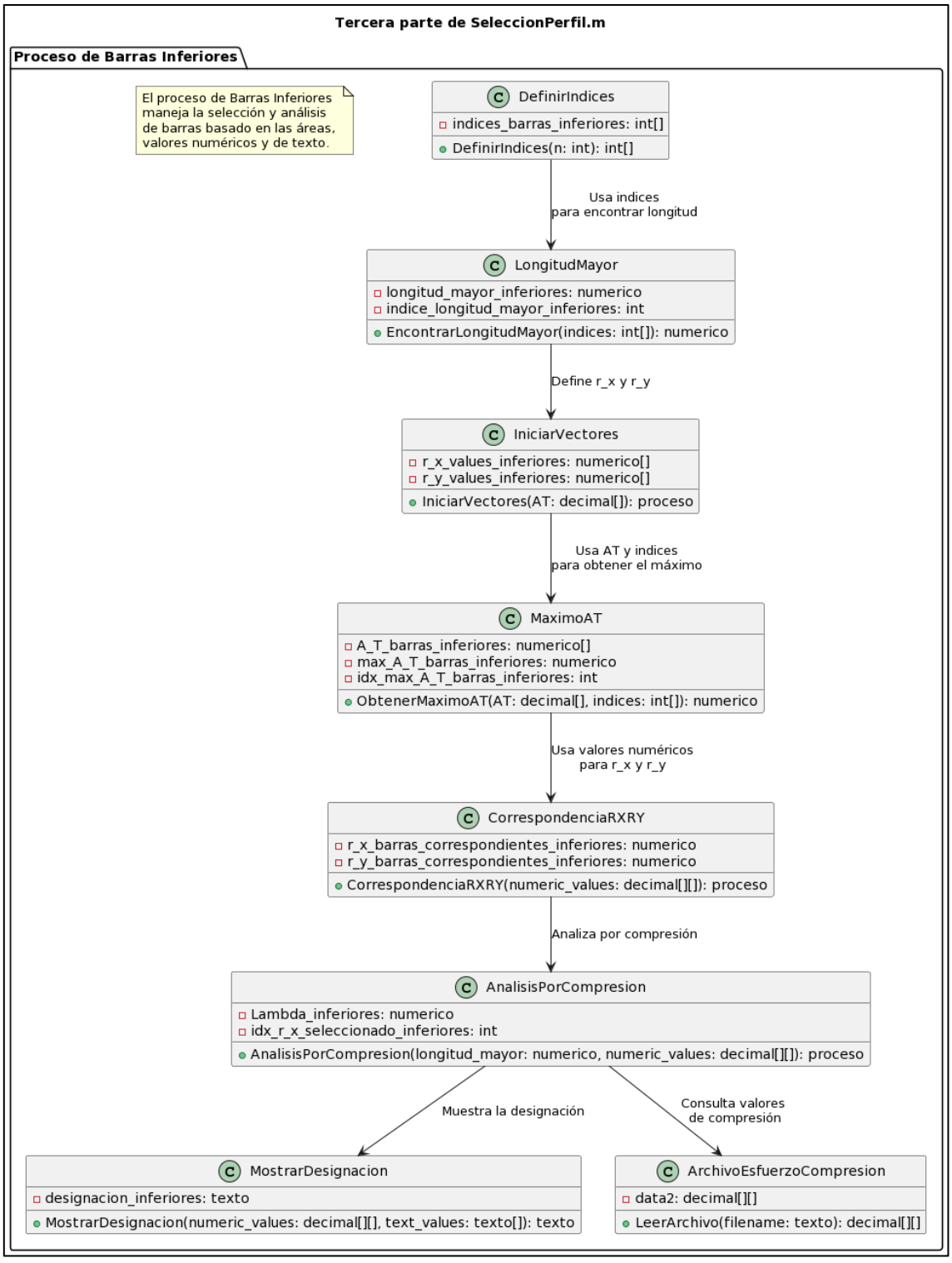


Figura 19

Cuarta Parte del Código SeleccionPerfil.m

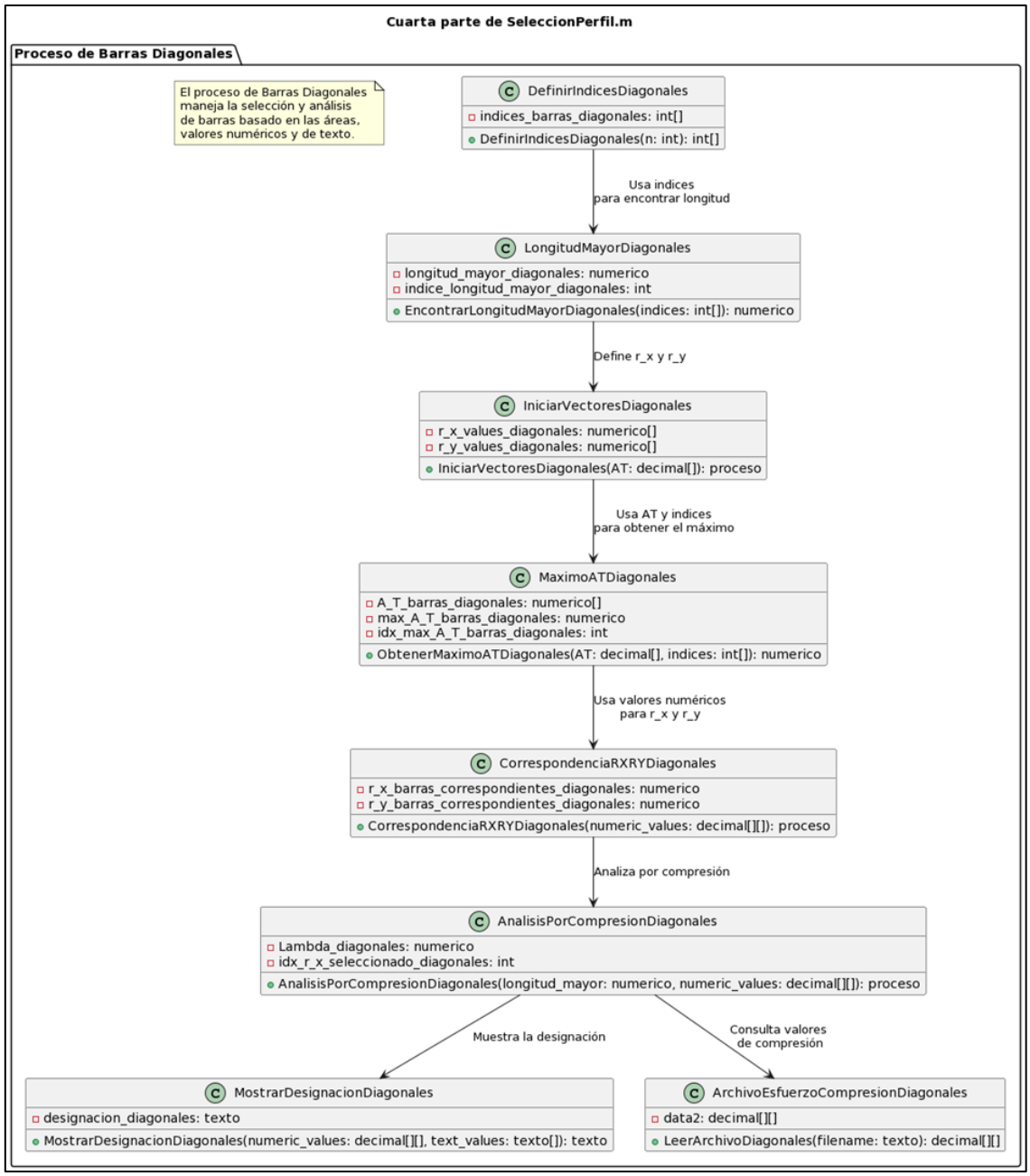


Figura 20

Quinta Parte del Código SeleccionPerfil.m

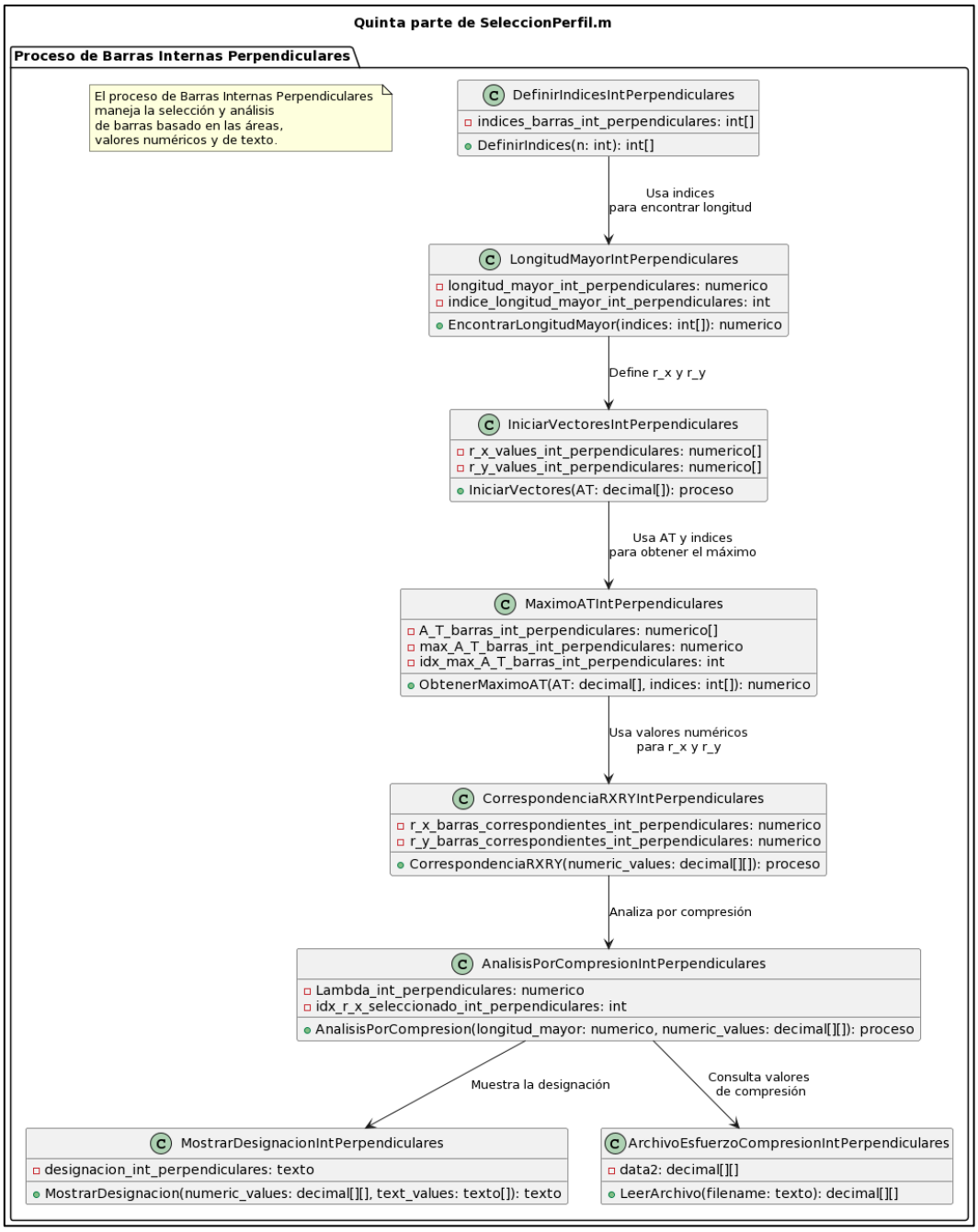


Figura 21

Sexta Parte del Código SeleccionPerfil.m

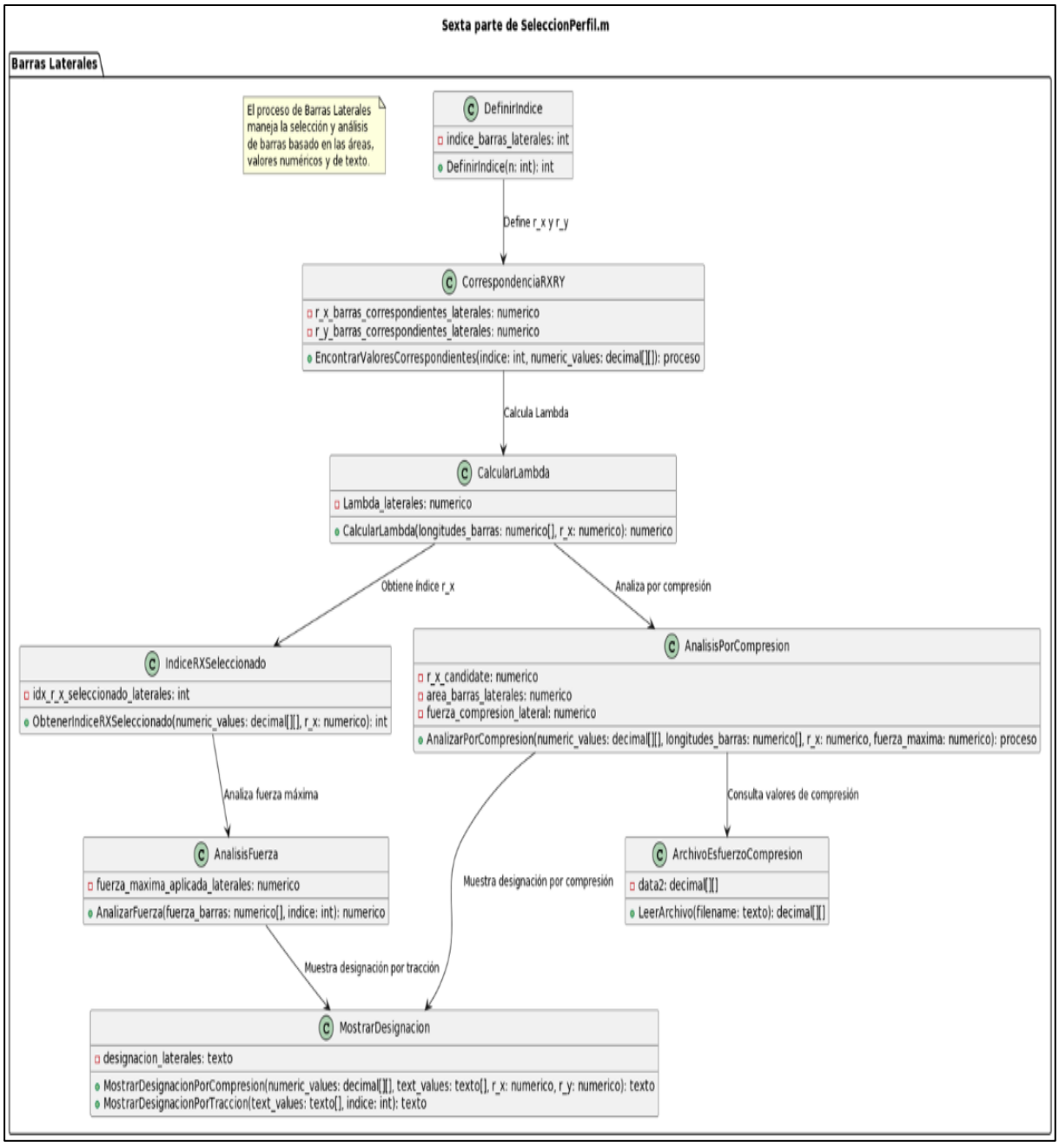
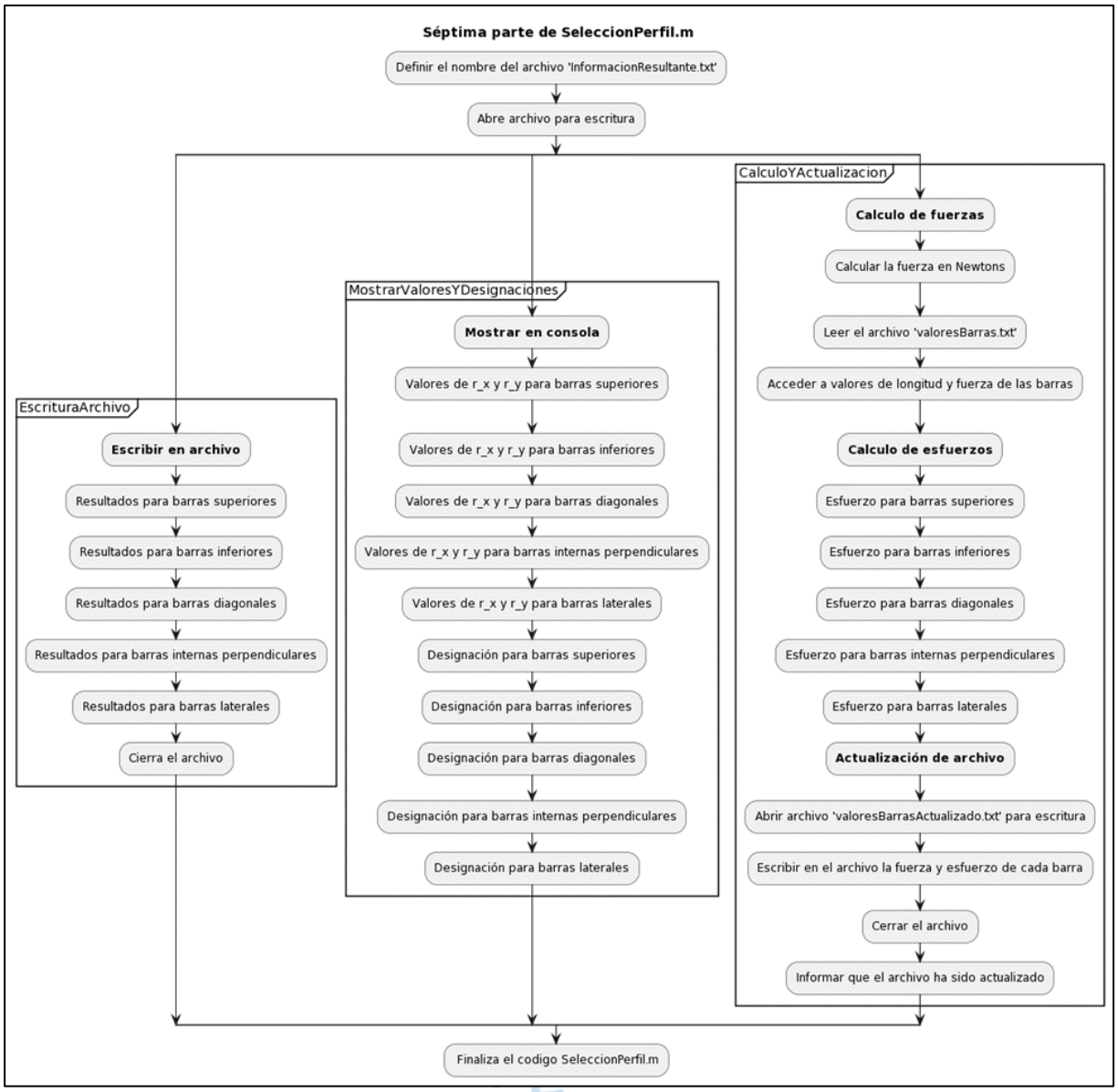


Figura 22

Séptima Parte del Código SeleccionPerfil.m



Capítulo 3
Casos prácticos en MATLAB

Para cada caso se ingresarán los datos en el Command Window, en el primer código "Tijeral" los datos a ingresar serán la longitud y su traslape correspondiente, en este primer ejemplo usaremos una longitud "L" y un traslape de calamina a calamina normado de "t". Después de esto el programa te dirá cuál es la pendiente de cada lado del tijeral y a la vez cual es largo más apto de las calaminas con respecto a la pendiente, y así nos dará los espacios (número de divisiones "n_divisiones") que habrá en casa lado para el tijeral. Luego de esto nos pedirá el largo de la división que viene a ser el espacio entre columnas donde se apoyará el próximo tijeral en una construcción a lo largo del terreno. El programa te pedirá que cantidad de nodos deseas restringir, para todos nuestros casos restringiremos solo 2, y serán los nodos 1 y el nodo que se da por la ecuación "2*n_divisiones + 1", para este último nodo no será restringido en el eje x para suponer los efectos por la dilatación.

Luego se generarán archivos que se usarán para ejecutar el siguiente código "Truss_TIJERAL" y dichos archivos inician con Datos_elementos requeridos" y otros archivos que luego servirán como referencia y base de datos. A continuación, se mostrará un esquema del funcionamiento de esta parte del codigo Truss_TIJERAL

El programa solicita el nombre Base de todos los archivos que necesita. Por ejemplo, si se ingresa el nombre de Datos el programa leerá los siguientes archivos:

- **Datos_**Info.txt
- **Datos_**Barras.txt
- **Datos_**Fuerzas.txt
- **Datos_**Nodos.txt
- **Datos_**Restricciones.txt

Después de los análisis genera un archivo llamado Datos_Resultados.txt

Tabla 6

Datos_Info.txt

Numero de barras	Número de nodos	Número de nodos con fuerzas	Número de nodos con restricciones

Sólo contiene estos cuatro datos.

Tabla 7

Datos_Barras.txt

Número de barra	Nodo I de la barra	Nodo J de la barra	Área de la barra	Módulo de elasticidad

Estos datos se colocan por cada una de las barras.

Tabla 8

Datos_Nodos.txt

Número de nodo	Coordenada X	Coordenada Y

Estos datos se colocan por cada uno de los nodos.

Tabla 9

Datos_Fuerzas.txt

Número de nodo	Fuerza en X	Fuerza en Y

Estos datos se colocan por cada una de las fuerzas.

Tabla 10

Datos_Restricciones.txt

Número de nodo	Restricción en X si vale 1, Si vale 0 está libre.	Restricción en Y si vale 1, Si vale 0 está libre.

Estos datos se colocan por cada una de las restricciones.

Por último, para el código final llamado “SeleccionPerfil” simplemente se ingresa el valor del número de divisiones obtenidos de la ejecución del primer código. Para ejecutar lo mencionado anteriormente se correrá el código de MATLAB en función al modelo propuesto, se realizarán 3 casos detallados:

3.1 Caso 1

3.1.1 Prueba y simulación del caso 1

3.1.1.1 Ingreso de datos en código Tijeral. Para este caso usaremos una longitud en metros “L” igual 15 y un traslape “t” igual a 0.15, luego de esto se ingresa el valor del largo de la división en metros “largo_division” igual a 6.

Figura 23

Command Window de código Tijeral de 15 metros.

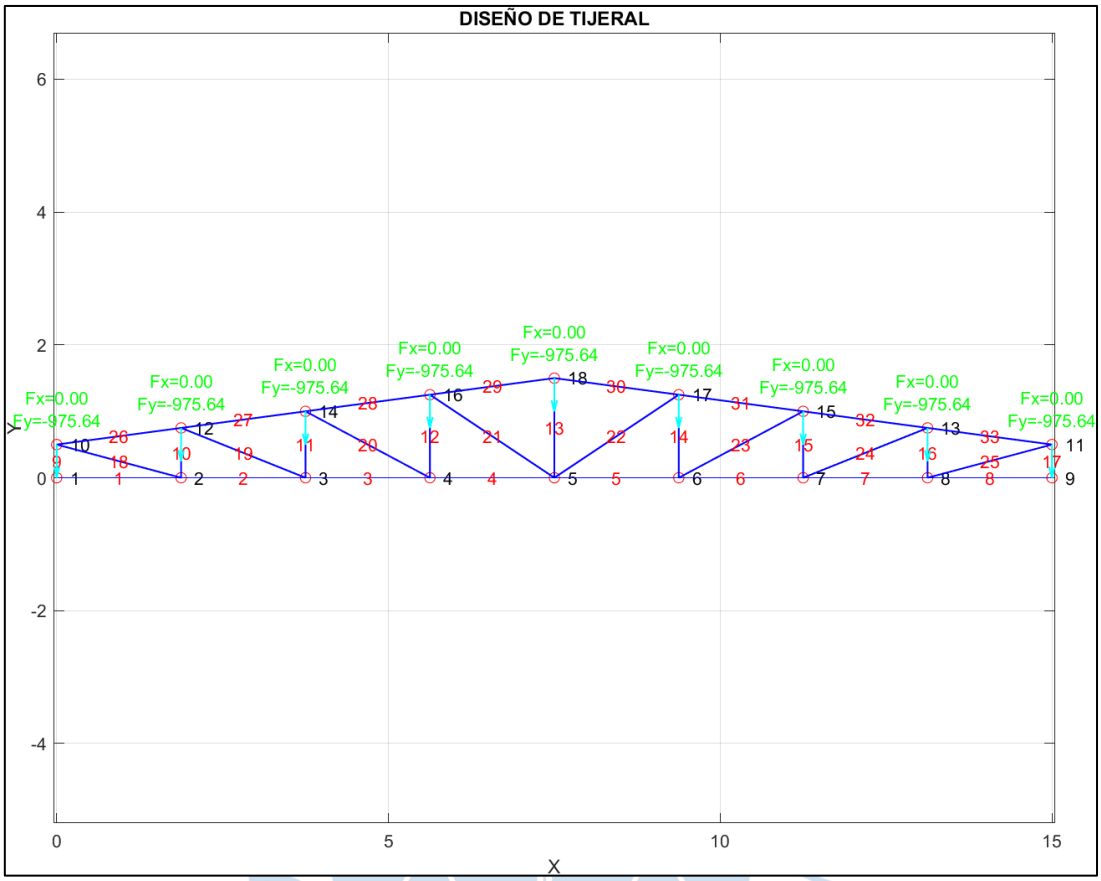
```
MATLAB Command Window                                          Page 1

Ingresa el valor de L en metros = 15
Ingresa el valor de t en metros = .15
El resultado de dividir la 7.57 entre la plancha P_1 resulta el numero de calaminas
N_1 igual a 5
El resultado de dividir la 7.57 entre la plancha P_2 resulta el numero de calaminas
N_2 igual a 4
El resultado de dividir la 7.57 entre la plancha P_3 resulta el numero de calaminas
N_3 igual a 4
N_1: 5
N_2: 4
N_3: 4
El valor de multiplicar N_1 por P_1 nos da como resultado la longitud final Lf_1
igual a 8.25
El valor de multiplicar N_2 por P_2 nos da como resultado la longitud final Lf_2
igual a 7.80
El valor de multiplicar N_3 por P_3 nos da como resultado la longitud final Lf_3
igual a 9.00
Lf_1: 8.25
Lf_2: 7.80
Lf_3: 9.00
El valor más cercano y mayor a 7.57 es 7.80
Por lo tanto, se utilizará la calamina P_2
El número de calaminas y espacios por lado a usar será 4
Ingresa el valor de largo_division en metros  = 6
Archivo "Datos_Fuerzas.txt" generado exitosamente.
Ingresa la cantidad de nodos a los que desea aplicar restricciones: 2
Ingresa el nodo 1 al que desea aplicar restricciones: 1
Ingresa el nodo 2 al que desea aplicar restricciones: 9
Ingresa la restricción en X para el nodo 1 (0 = desplazamiento, 1 = fijo): 1
Ingresa la restricción en Y para el nodo 1 (0 = desplazamiento, 1 = fijo): 1
Ingresa la restricción en X para el nodo 9 (0 = desplazamiento, 1 = fijo): 0
Ingresa la restricción en Y para el nodo 9 (0 = desplazamiento, 1 = fijo): 1
Archivo "Datos_Nodos.txt" generado exitosamente.
Archivo "Informacion_Barras.txt" generado exitosamente.
Archivo "Datos_Barras.txt" generado exitosamente.
Archivo "valoresBarras.txt" generado exitosamente.
Archivo "Datos_Info.txt" generado exitosamente.
>>
```

Luego de haber ejecutado el código, se imprimirá una imagen que te muestra el plano del tijeral con sus fuerzas en cada nodo de la parte superior, es necesario que estas fuerzas que se están aplicando, está conformada por la carga viva, la carga muerta y la carga del viento:

Figura 24

Imagen 2D del tijeral de 15 metros.



3.1.1.2 Ingreso de datos en código Truss_TIJERAL. Para esta ejecución de código se ingresa la denominación Datos y tomará los archivos generados anteriormente iniciando con Datos.

Figura 25

Command Window de Truss_tijeral de 15 metros.

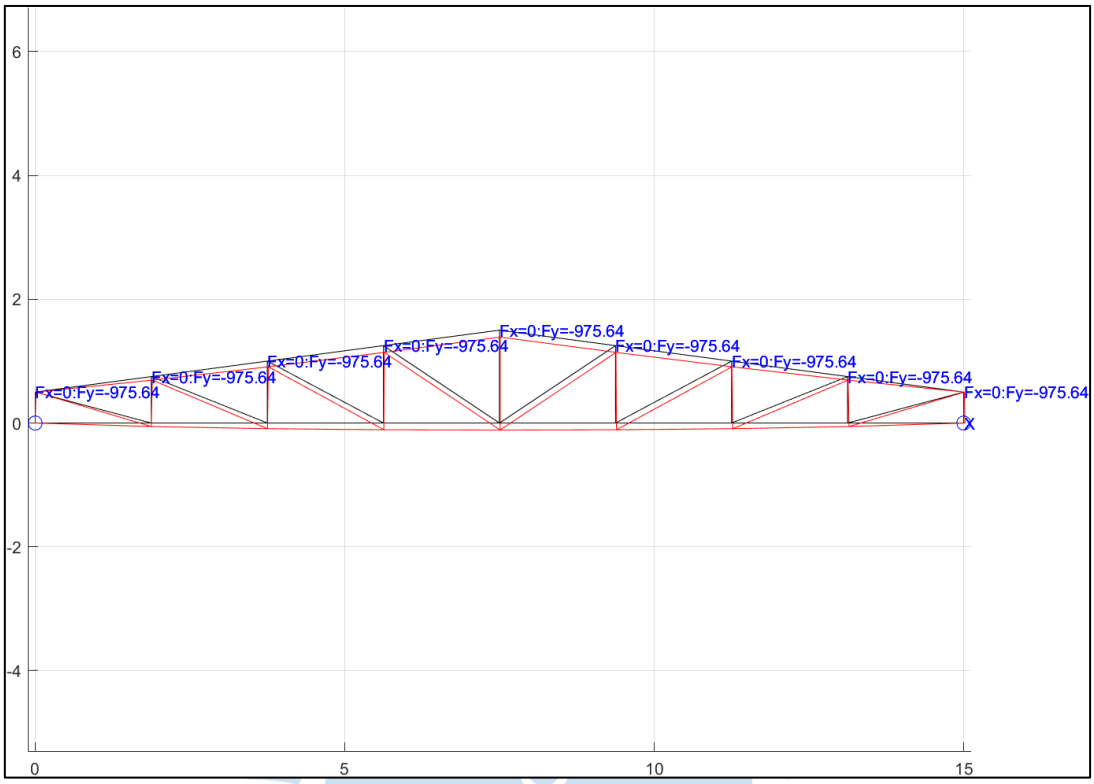
```
MATLAB Command Window                                    Page 1

Ingresar nombre del archivo de Datos = Datos
Archivo "valoresBarras.txt" actualizado con la fuerza de cada barra.
>>
```

Luego de haber ejecutado el código, se imprimirá una imagen que te muestra el plano del tijeral con su respectivo desplazamiento, la representación del color rojo es el desplazamiento producido por la fuerza a la que se somete el tijeral.

Figura 26

Plano 2D de código de Truss_tijeral de 15 metros.



3.1.1.3 Ingreso de datos en código SeleccionPerfil. Para esta ejecución de código se ingresa el número de divisiones “n_divisiones” obtenido en la ejecución del primer código “Tijeral”.

Figura 27

Command Window de SeleccionPerfil de 15 metros.

```
MATLAB Command Window                                    Page 1

Ingresa el valor de n_divisiones = 4
```

3.1.2 Resultados obtenidos del caso 1

Una vez ejecutado el ultimo código obtenemos dos archivos, una tabla Excel que nos muestra la información de cada barra del tijeral “ValoresBarrasActualizado” como su número de barra, longitud, fuerzas y esfuerzo y el archivo de texto que nos muestra información necesaria para el diseño del tijeral con sus respectivos ángulos para cada zona llamado “InformacionResultante”.

Tabla 11

Resultados en MATLAB de tijeral de 15 metros.

N° Barra	Longitud (m)	Fuerza (kg)	Fuerza (N)	Esfuerzo (Pa)
1	1.875	0	0	0
2	1.875	8536	83746	69074974
3	1.875	10975	107674	88810680
4	1.875	10975	107674	88810680
5	1.875	10975	107674	88810680
6	1.875	10975	107674	88810680
7	1.875	8536	83746	69074974
8	1.875	0	0	0
9	0.500	-4390	43069	142331883
10	0.750	-2276	22332	58278619
11	1.000	-975	9571	24976587
12	1.250	0	0	0
13	1.500	1626	15951	41627731
14	1.250	0	0	0
15	1.000	-975	9571	24976587
16	0.750	-2276	22332	58278619
17	0.500	-4390	43069	0
18	1.941	8835	86673	58408934
19	2.019	2626	25770	17366919
20	2.125	0	0	0
21	2.254	-1465	14378	9689812
22	2.254	-1465	14378	9689812
23	2.125	0	0	0
24	2.019	2626	25770	17366919
25	1.941	8835	86673	58408934
26	1.892	-8612	84487	69686278
27	1.892	-11073	108626	89596597

28	1.892	-11073	108626	89596597
29	1.892	-9842	96557	79641437
30	1.892	-9842	96557	79641437
31	1.892	-11073	108626	89596597
32	1.892	-11073	108626	89596597
33	1.892	-8612	84487	69686278

Este archivo te muestra la máxima fuerza de compresión a la que se somete el elemento de acuerda a la zona de análisis y al hacerlos cálculos pertinentes halla su radio de giro adecuado para dicha fuerza y así encuentra el tipo de ángulo más adecuado para la fuerza a la cual se somete.

Figura 28

Resultado de los perfiles para tijeral de 15 metros.

Tijeral.m | Truss_TIJERAL.m | SeleccionPerfil.m | valoresBarrasActualizado.txt | InformacionResultante.txt

```
RESULTADOS PARA LAS BARRAS SUPERIORES:
Fuerza máxima aplicada: 11073.080000 kg
Fuerza de compresion dado el angulo seleccioando: 11881.520000 kg
Valores seleccionados para las barras superiores: r_x = 1.547 cm, r_y = 3.697 cm
Ángulo seleccionado por compresión: 2 x 2 x 1/4

RESULTADOS PARA LAS BARRAS INFERIORES:
Fuerza máxima aplicada: 10975.950000 kg
Fuerza de compresion dado el angulo seleccionado: 12124.000000 kg
Valores seleccionados para las barras inferiores: r_x = 1.547 cm, r_y = 3.697 cm
Ángulo seleccionado por compresión: 2 x 2 x 1/4

RESULTADOS PARA LAS BARRAS DIAGONALES:
Fuerza máxima aplicada: 8835.170000 kg
Fuerza de compresion dado el angulo seleccionado: 10387.300000 kg
Valores seleccionados para las barras diagonales: r_x = 1.527 cm, r_y = 3.637 cm
Ángulo seleccionado por compresión: 2 x 2 x 5/16

RESUSLTADOS PARA LAS BARRAS INTERNAS PERPENDICULARES:
Fuerza máxima aplicada: 2276.490000 kg
Fuerza de compresion dado el angulo seleccionado: 2452.480000 kg
Valores seleccionados para las barras internas perpendiculares: r_x = 0.978 cm, r_y = 2.465 cm
Ángulo seleccionado por compresión: 1 1/4 x 1 1/4 x 1/8

RESULTADO PARA LAS BARRAS LATERALES:
Fuerza máxima aplicada: 4390.380000 kg
Fuerza de compresion dado el angulo seleccionado: 5265.240000 kg
Valores seleccionados para las barras laterales: r_x = 0.772 cm, r_y = 1.948 cm
Ángulo seleccionado por compresión: 1x1x1/8
```

3.2 Caso 2

3.2.1 Prueba y simulación del caso 2

3.2.1.1 Ingreso de datos en código Tijeral. Para este caso usaremos una longitud en metros “L” igual 20 y un traslape “t” igual a 0.15, luego de esto se ingresa el valor del largo de la división en metros “largo_division” igual a 6.

Figura 29

Command Window de código Tijeral de 20 metros.

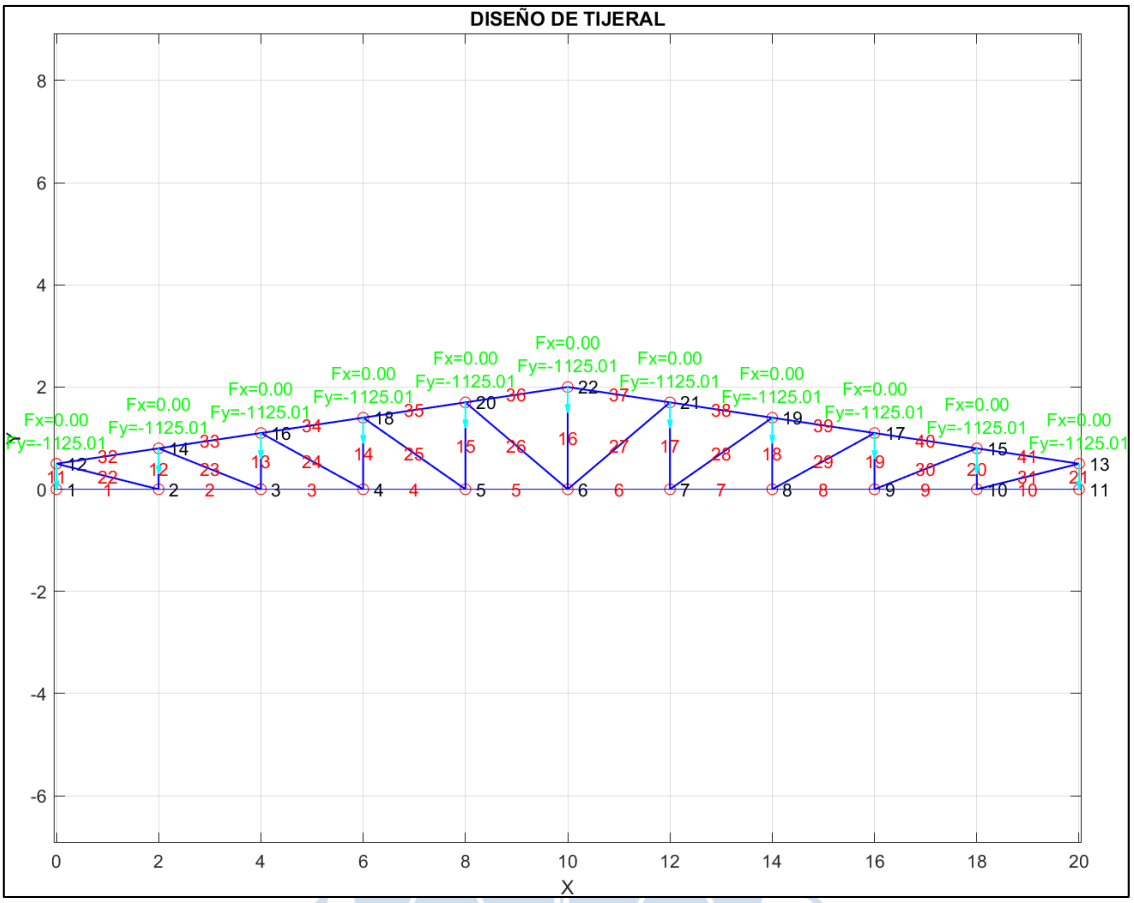
```
MATLAB Command Window                                          Page 1

Ingresa el valor de L en metros = 20
Ingresa el valor de t en metros = .15
El resultado de dividir la 10.12 entre la plancha P_1 resulta el numero de ↙
calaminas N_1 igual a 7
El resultado de dividir la 10.12 entre la plancha P_2 resulta el numero de ↙
calaminas N_2 igual a 6
El resultado de dividir la 10.12 entre la plancha P_3 resulta el numero de ↙
calaminas N_3 igual a 5
N_1: 7
N_2: 6
N_3: 5
El valor de multiplicar N_1 por P_1 nos da como resultado la longitud final Lf_1 ↙
igual a 11.55
El valor de multiplicar N_2 por P_2 nos da como resultado la longitud final Lf_2 ↙
igual a 11.70
El valor de multiplicar N_3 por P_3 nos da como resultado la longitud final Lf_3 ↙
igual a 11.25
Lf_1: 11.55
Lf_2: 11.70
Lf_3: 11.25
El valor más cercano y mayor a 10.12 es 11.25
Por lo tanto, se utilizará la calamina P_3
El número de calaminas y espacios por lado a usar será 5
Ingresa el valor de largo_division en metros  = 6
Archivo "Datos_Fuerzas.txt" generado exitosamente.
Ingrese la cantidad de nodos a los que desea aplicar restricciones: 2
Ingrese el nodo 1 al que desea aplicar restricciones: 1
Ingrese el nodo 2 al que desea aplicar restricciones: 11
Ingrese la restricción en X para el nodo 1 (0 = desplazamiento, 1 = fijo): 1
Ingrese la restricción en Y para el nodo 1 (0 = desplazamiento, 1 = fijo): 1
Ingrese la restricción en X para el nodo 11 (0 = desplazamiento, 1 = fijo): 0
Ingrese la restricción en Y para el nodo 11 (0 = desplazamiento, 1 = fijo): 1
Archivo "Datos_Nodos.txt" generado exitosamente.
Archivo "Informacion_Barras.txt" generado exitosamente.
Archivo "Datos_Barras.txt" generado exitosamente.
Archivo "valoresBarras.txt" generado exitosamente.
Archivo "Datos_Info.txt" generado exitosamente.
>>
```

Luego de haber ejecutado el código, se imprimirá una imagen que te muestra el plano del tijeral con sus fuerzas en cada nodo de la parte superior, es necesario que estas fuerzas que se están aplicando, está conformada por la carga viva, la carga muerta y la carga del viento:

Figura 30

Imagen 2D del tijeral de 20 metros.



3.2.1.2 Ingreso de datos en código Truss_TIJERAL. Para esta ejecución de código se ingresa la denominación Datos y tomará los archivos generados anteriormente iniciando con Datos.

Figura 31

Command Window de Truss_tijeral de 20 metros.

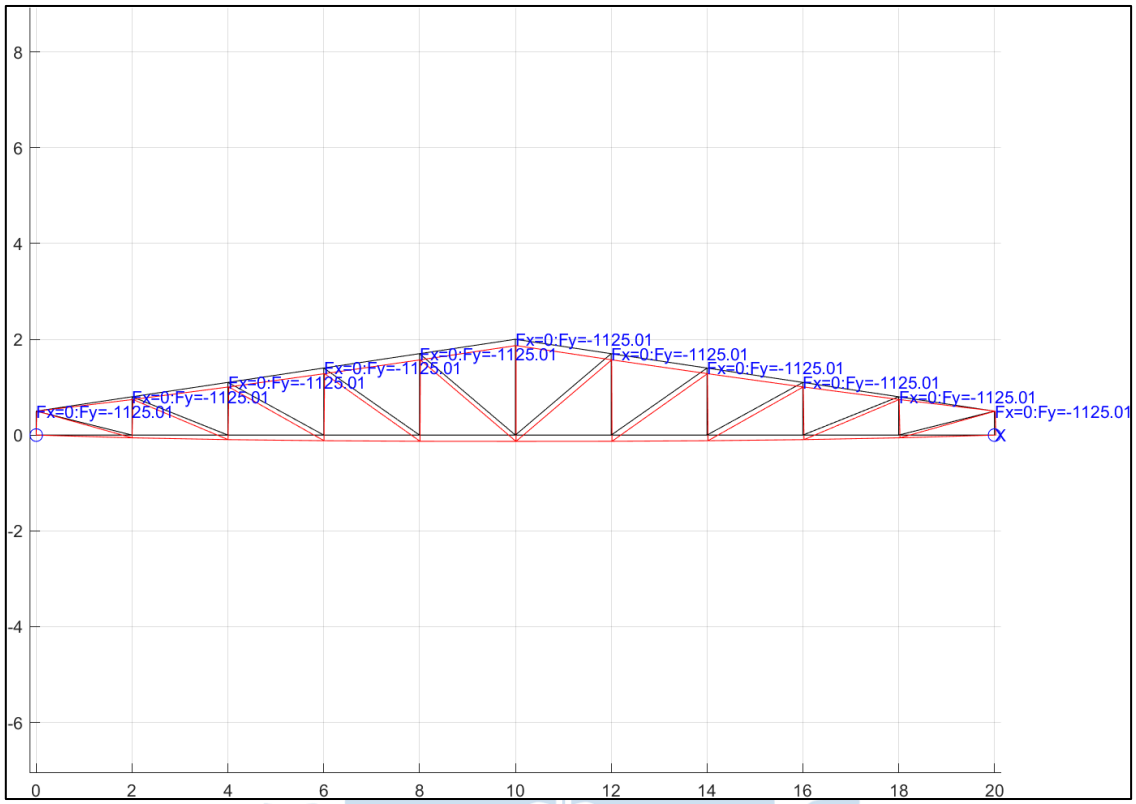
```
MATLAB Command Window                                    Page 1

Ingresar nombre del archivo de Datos = Datos
Archivo "valoresBarras.txt" actualizado con la fuerza de cada barra.
>>
```

Luego de haber ejecutado el código, se imprimirá una imagen que te muestra el plano del tijeral con su respectivo desplazamiento, la representación del color rojo es el desplazamiento producido por la fuerza a la que se somete el tijeral.

Figura 32

Plano 2D de código de Truss_tijeral de 20 metros.



3.2.1.3 Ingreso de datos en código SeleccionPerfil. Para esta ejecución de código se ingresa el número de divisiones “n_divisiones” obtenido en la ejecución del primer código “Tijeral”.

Figura 33

Command Window de SeleccionPerfil de 20 metros.

```
MATLAB Command Window                                    Page 1

Ingresa el valor de n_divisiones = 5
```

3.2.2 Resultados obtenidos del caso 2

Una vez ejecutado el ultimo código obtenemos dos archivos, una tabla Excel que nos muestra la información de cada barra del tijeral “ValoresBarrasActualizado” como su número de barra, longitud, fuerzas y esfuerzo y el archivo de texto que nos muestra información necesaria para el diseño del tijeral con sus respectivos ángulos para cada zona llamado “InformacionResultante”.

Tabla 12

Resultados en MATLAB de tijeral de 20 metros.

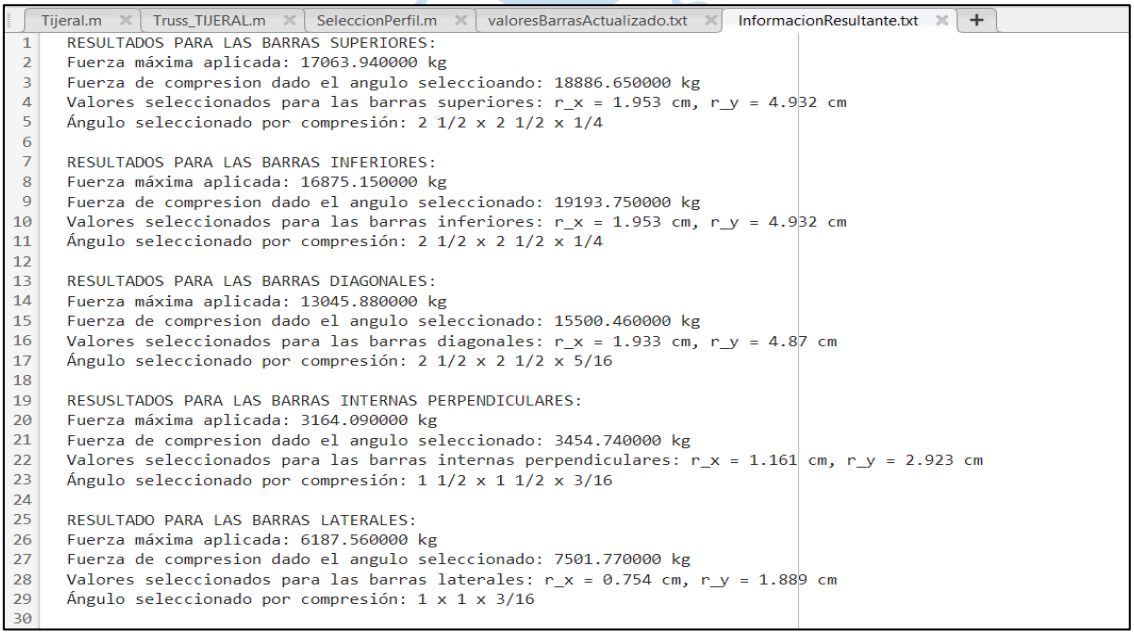
N° Barra	Longitud (m)	Fuerza (kg)	Fuerza (N)	Esfuerzo (Pa)
1	2.000	0	0	0
2	2.000	12656	124158	80858932
3	2.000	16363	160528	104544892
4	2.000	16875	165545	107811931
5	2.000	15882	155807	101470027
6	2.000	15882	155807	101470027
7	2.000	16875	165545	107811931
8	2.000	16363	160528	104544892
9	2.000	12656	124158	80858932
10	2.000	0	0	0
11	0.500	-6187	60699	138363263
12	0.800	-3164	31039	45821852
13	1.100	-1482	14547	21476137
14	1.400	-281	2759	4073018
15	1.700	694	6816	10062852
16	2.000	3093	30349	44803634
17	1.700	694	6816	10062852
18	1.400	-281	2759	4073018
19	1.100	-1482	14547	21476137
20	0.800	-3164	31039	45821852
21	0.500	-6187	60699	0
22	2.062	13045	127980	67703582
23	2.154	3993	39171	20722334
24	2.283	583	5725	3028733
25	2.441	-1211	11886	6288249
26	2.625	-2388	23430	12395329
27	2.625	-2388	23430	12395329

28	2.441	-1211	11886	6288249
29	2.283	583	5725	3028733
30	2.154	3993	39171	20722334
31	2.062	13045	127980	67703582
32	2.022	-12797	125547	81763522
33	2.022	-16546	162324	105714489
34	2.022	-17063	167397	109018073
35	2.022	-16060	157550	102605252
36	2.022	-14219	139497	90848394
37	2.022	-14219	139497	90848394
38	2.022	-16060	157550	102605252
39	2.022	-17063	167397	109018073
40	2.022	-16546	162324	105714489
41	2.022	-12797	125547	81763522

Este archivo te muestra la máxima fuerza de compresión a la que se somete el elemento de acuerda a la zona de análisis y al hacerlos cálculos pertinentes halla su radio de giro adecuado para dicha fuerza y así encuentra el tipo de ángulo más adecuado para la fuerza a la cual se somete.

Figura 34

Resultado de los perfiles para tijeral de 20 metros.



3.3 Caso 3

3.3.1 Prueba y simulación del caso 3

3.3.1.1 Ingreso de datos en código Tijeral. Para este caso usaremos una longitud en metros “L” igual 27 y un traslape “t” igual a 0.15, luego de esto se ingresa el valor del largo de la división en metros “largo_division” igual a 6.

Figura 35

Command Window de código Tijeral de 27 metros.

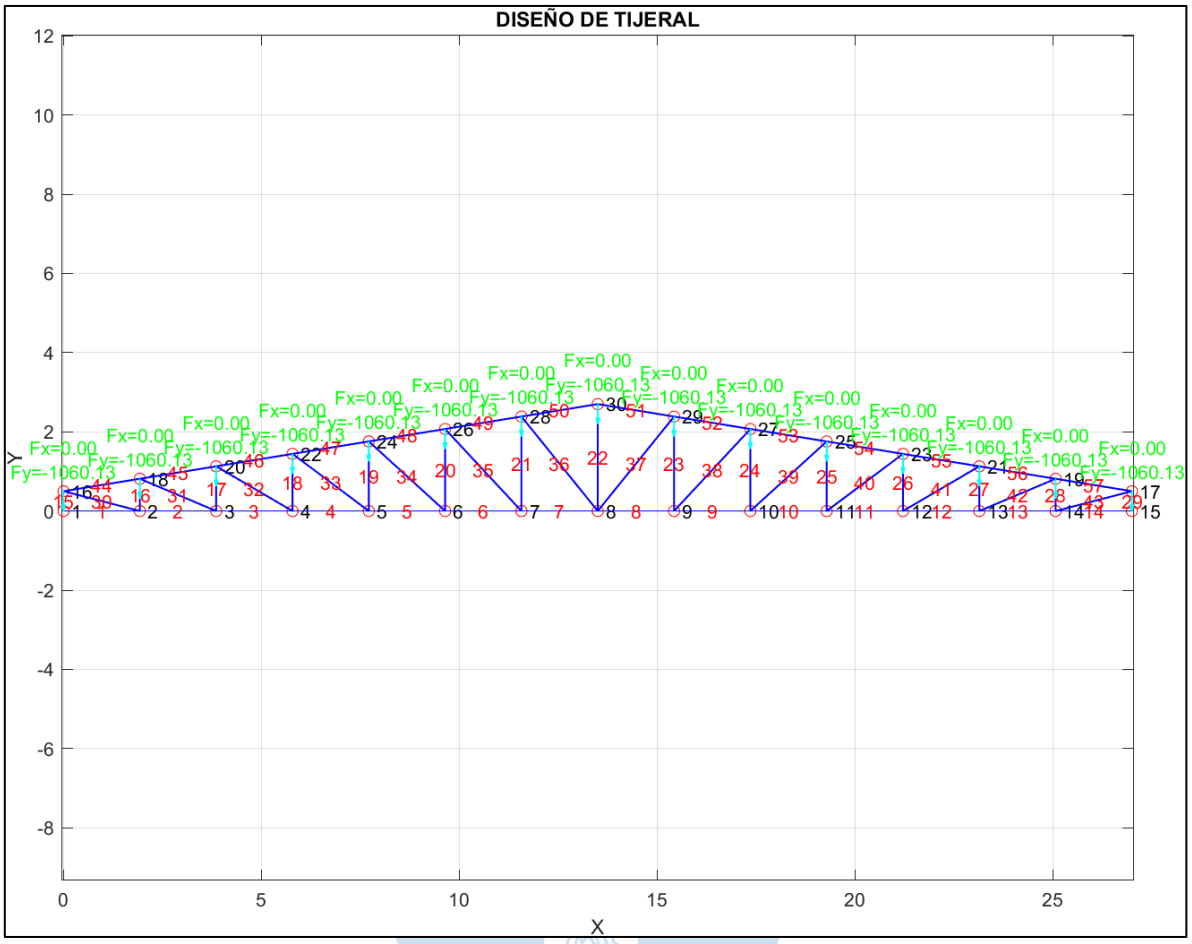
```
MATLAB Command Window                                          Page 1

Ingresa el valor de L en metros = 27
Ingresa el valor de t en metros = .15
El resultado de dividir la 13.68 entre la plancha P_1 resulta el numero de ↙
calaminas N_1 igual a 9
El resultado de dividir la 13.68 entre la plancha P_2 resulta el numero de ↙
calaminas N_2 igual a 8
El resultado de dividir la 13.68 entre la plancha P_3 resulta el numero de ↙
calaminas N_3 igual a 7
N_1: 9
N_2: 8
N_3: 7
El valor de multiplicar N_1 por P_1 nos da como resultado la longitud final Lf_1 ↙
igual a 14.85
El valor de multiplicar N_2 por P_2 nos da como resultado la longitud final Lf_2 ↙
igual a 15.60
El valor de multiplicar N_3 por P_3 nos da como resultado la longitud final Lf_3 ↙
igual a 15.75
Lf_1: 14.85
Lf_2: 15.60
Lf_3: 15.75
El valor más cercano y mayor a 13.68 es 14.85
Por lo tanto, se utilizará la calamina P_1
El número de calaminas y espacios por lado a usar será 7
Ingresa el valor de largo_division en metros  = 6
Archivo "Datos_Fuerzas.txt" generado exitosamente.
Ingresa la cantidad de nodos a los que desea aplicar restricciones: 2
Ingresa el nodo 1 al que desea aplicar restricciones: 1
Ingresa el nodo 2 al que desea aplicar restricciones: 15
Ingresa la restricción en X para el nodo 1 (0 = desplazamiento, 1 = fijo): 1
Ingresa la restricción en Y para el nodo 1 (0 = desplazamiento, 1 = fijo): 1
Ingresa la restricción en X para el nodo 15 (0 = desplazamiento, 1 = fijo): 0
Ingresa la restricción en Y para el nodo 15 (0 = desplazamiento, 1 = fijo): 1
Archivo "Datos_Nodos.txt" generado exitosamente.
Archivo "Informacion_Barras.txt" generado exitosamente.
Archivo "Datos_Barras.txt" generado exitosamente.
Archivo "valoresBarras.txt" generado exitosamente.
Archivo "Datos_Info.txt" generado exitosamente.
>>
```

Luego de haber ejecutado el código, se imprimirá una imagen que te muestra el plano del tijeral con sus fuerzas en cada nodo de la parte superior, es necesario que estas fuerzas que se están aplicando, está conformada por la carga viva, la carga muerta y la carga del viento:

Figura 36

Imagen 2D del tijeral de 27 metros.



3.3.1.2 Ingreso de datos en código Truss_TIJERAL. Para esta ejecución de código se ingresa la denominación Datos y tomará los archivos generados anteriormente iniciando con Datos.

Figura 37

Command Window de Truss_tijeral de 27 metros

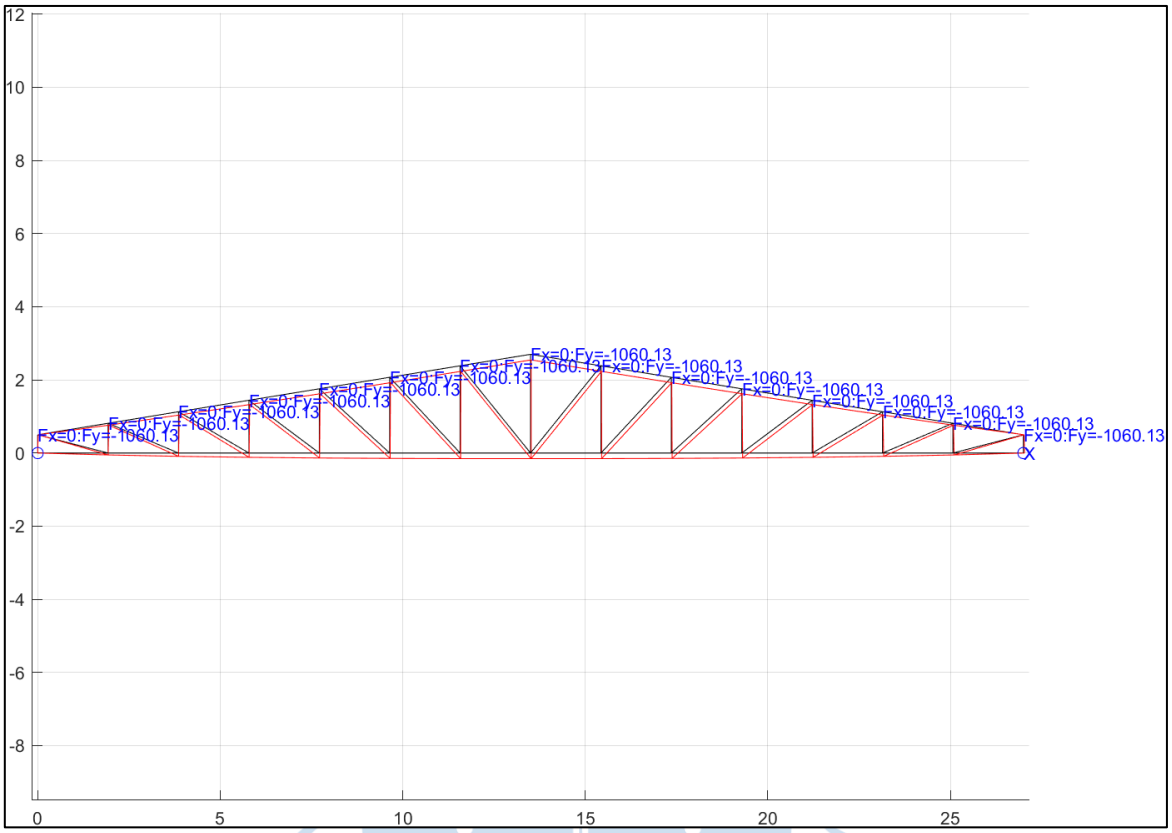
```
MATLAB Command Window                                    Page 1

Ingresar nombre del archivo de Datos = Datos
Archivo "valoresBarras.txt" actualizado con la fuerza de cada barra.
>>
```

Luego de haber ejecutado el código, se imprimirá una imagen que te muestra el plano del tijeral con su respectivo desplazamiento, la representación del color rojo es el desplazamiento producido por la fuerza a la que se somete el tijeral.

Figura 38

Plano 2D de código de Truss_tijeral de 27 metros.



3.3.1.3 Ingreso de datos en código SeleccionPerfil. Para esta ejecución de código se ingresa el número de divisiones "n_divisiones" obtenido en la ejecución del primer código "Tijeral".

Figura 39

Command Window de SeleccionPerfil de 27 metros.

```
MATLAB Command Window                                    Page 1

Ingresa el valor de n_divisiones = 7
```

3.3.2 Resultados obtenidos del caso 3

Una vez ejecutado el ultimo código obtenemos dos archivos, una tabla Excel que nos muestra la información de cada barra del tijeral "ValoresBarrasActualizado" como su número de barra, longitud, fuerzas y esfuerzo y el archivo de texto que nos muestra información necesaria para el diseño del tijeral con sus respectivos ángulos para cada zona llamado "InformacionResultante".

Tabla 13

Resultados en MATLAB de tijeral de 27 metros.

N° Barra	Longitud (m)	Fuerza (kg)	Fuerza (N)	Esfuerzo (Pa)
1	1.929	0	0	0
2	1.929	16329	160195	84745882
3	1.929	21730	213179	112775247
4	1.929	23379	229352	121331115
5	1.929	23272	228305	120777639
6	1.929	22212	217907	115276815
7	1.929	20565	201744	106726395
8	1.929	20565	201744	106726395
9	1.929	22212	217907	115276815
10	1.929	23272	228305	120777639
11	1.929	23379	229352	121331115
12	1.929	21730	213179	112775247
13	1.929	16329	160195	84745882
14	1.929	0	0	0
15	0.500	-7950	77999	138173806
16	0.814	-4232	41522	3424850
17	1.129	-2280	22369	18450794
18	1.443	-964	9465	7807462
19	1.757	79	783	645854
20	2.071	965	9471	7811831
21	2.386	1769	17361	14320059
22	2.700	4979	48850	40292689
23	2.386	1769	17361	14320059
24	2.071	965	9471	7811831
25	1.757	79	783	645854
26	1.443	-964	9465	7807462
27	1.129	-2280	22369	18450794

28	0.814	-4232	41522	34248503
29	0.500	-7950	77999	0
30	1.992	16869	165489	72256477
31	2.093	5862	57512	25111381
32	2.235	1910	18739	8182181
33	2.409	-133	1306	570575
34	2.609	-1433	14064	6141112
35	2.830	-2417	23720	10356932
36	3.068	-3201	31409	13714082
37	3.068	-3201	31409	13714082
38	2.830	-2417	23720	10356932
39	2.609	-1433	14064	6141112
40	2.409	-133	1306	570575
41	2.235	1910	18739	8182181
42	2.093	5862	57512	25111381
43	1.992	16869	165489	72256477
44	1.954	-16544	162303	85861294
45	1.954	-22018	216005	114270489
46	1.954	-23687	232370	122928075
47	1.954	-23579	231314	122368942
48	1.954	-22505	220775	116794062
49	1.954	-20837	204419	108141510
50	1.954	-18796	184393	97547094
51	1.954	-18796	184393	97547094
52	1.954	-20837	204419	108141510
53	1.954	-22505	220775	116794062
54	1.954	-23579	231314	122368942
55	1.954	-23687	232370	122928075
56	1.954	-22018	216005	114270489
57	1.954	-16544	162303	85861294

Este archivo te muestra la máxima fuerza de compresión a la que se somete el elemento de acuerda a la zona de análisis y al hacerlos cálculos pertinentes halla su radio de giro adecuado para dicha fuerza y así encuentra el tipo de ángulo más adecuado para la fuerza a la cual se somete.

Figura 40

Resultado de los perfiles para tijeral de 27 metros.

Tijeral.m | Truss_TIJERAL.m | SeleccionPerfil.m | valoresBarrasActualizado.txt | InformacionResultante.txt

```
RESULTADOS PARA LAS BARRAS SUPERIORES:
Fuerza máxima aplicada: 23687.150000 kg
Fuerza de compresion dado el angulo seleccioando: 23817.780000 kg
Valores seleccionados para las barras superiores: r_x = 1.933 cm, r_y = 4.87 cm
Ángulo seleccionado por compresión: 2 1/2 x 2 1/2 x 5/16

RESULTADOS PARA LAS BARRAS INFERIORES:
Fuerza máxima aplicada: 23379.430000 kg
Fuerza de compresion dado el angulo seleccionado: 24384.870000 kg
Valores seleccionados para las barras inferiores: r_x = 1.933 cm, r_y = 4.87 cm
Ángulo seleccionado por compresión: 2 1/2 x 2 1/2 x 5/16

RESULTADOS PARA LAS BARRAS DIAGONALES:
Fuerza máxima aplicada: 16869.420000 kg
Fuerza de compresion dado el angulo seleccionado: 20383.670000 kg
Valores seleccionados para las barras diagonales: r_x = 2.342 cm, r_y = 5.907 cm
Ángulo seleccionado por compresión: 3 x 3 x 5/16

RESUSLTADOS PARA LAS BARRAS INTERNAS PERPENDICULARES:
Fuerza máxima aplicada: 4979.700000 kg
Fuerza de compresion dado el angulo seleccionado: 6062.000000 kg
Valores seleccionados para las barras internas perpendiculares: r_x = 1.547 cm, r_y = 3.697 cm
Ángulo seleccionado por compresión: 2 x 2 x 1/4

RESULTADO PARA LAS BARRAS LATERALES:
Fuerza máxima aplicada: 7950.980000 kg
Fuerza de compresion dado el angulo seleccionado: 9596.500000 kg
Valores seleccionados para las barras laterales: r_x = 0.737 cm, r_y = 1.833 cm
Ángulo seleccionado por compresión: 1 x 1 x 1/4
```

Capítulo 4
Desarrollo de los casos prácticos en SolidWorks

En el presente capítulo, se abordará minuciosamente el proceso de modelado y construcción de los tijerales, una parte fundamental en la estructura de este proyecto. Este proceso de diseño y modelado se desarrolló con gran precisión utilizando la versión 2021 mediante el software SolidWorks y SolidWorks Simulation, una plataforma altamente reconocida en la industria para la creación de modelos tridimensionales. Durante este proceso, se aplicaron metodologías avanzadas de diseño asistido por computadora con el objetivo de asegurar la exactitud y eficacia en la representación digital de los tijerales.

Para el desarrollo del tijeral en los diferentes casos propuestos, se plantea la utilización de canales en forma de "C" conformados a partir de ángulos en forma de "L". Estos ángulos se determinaron a partir de los datos obtenidos en el capítulo anterior mediante el uso del software MATLAB.

En el análisis estático, se empleó la funcionalidad de SolidWorks Simulation, donde se optó por el empleo de un material primordial, específicamente el acero A36. Seguidamente, se describen las restricciones y las cargas externas, las cuales se ajustaron manualmente. Posteriormente, se procedió a generar un mallado adecuado mediante la técnica de elementos finitos para llevar a cabo el análisis. Como resultado de este proceso se obtuvieron gráficos que se representan las tensiones y los desplazamientos en la estructura.

El propósito fundamental de este proyecto es determinar las fuerzas presentes en cada una de las barras en los diversos casos planteados. Estos resultados se obtuvieron a través de dos programas informáticos: MATLAB y SolidWorks. La elección de estos programas se basa en su capacidad para proporcionar datos precisos y confiables para el análisis de las estructuras de los tijerales.

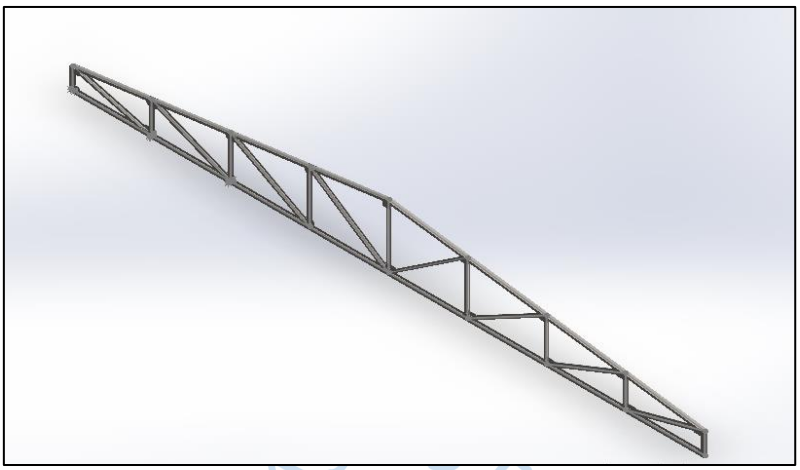
4.1 Caso 1

4.1.1 Diseño del caso 1

En la primera situación propuesta, se ha planteado que la dimensión de la base del tijeral sea de 15 metros. Por lo tanto, en la figura 41 se presenta el modelo tridimensional correspondiente.

Figura 41

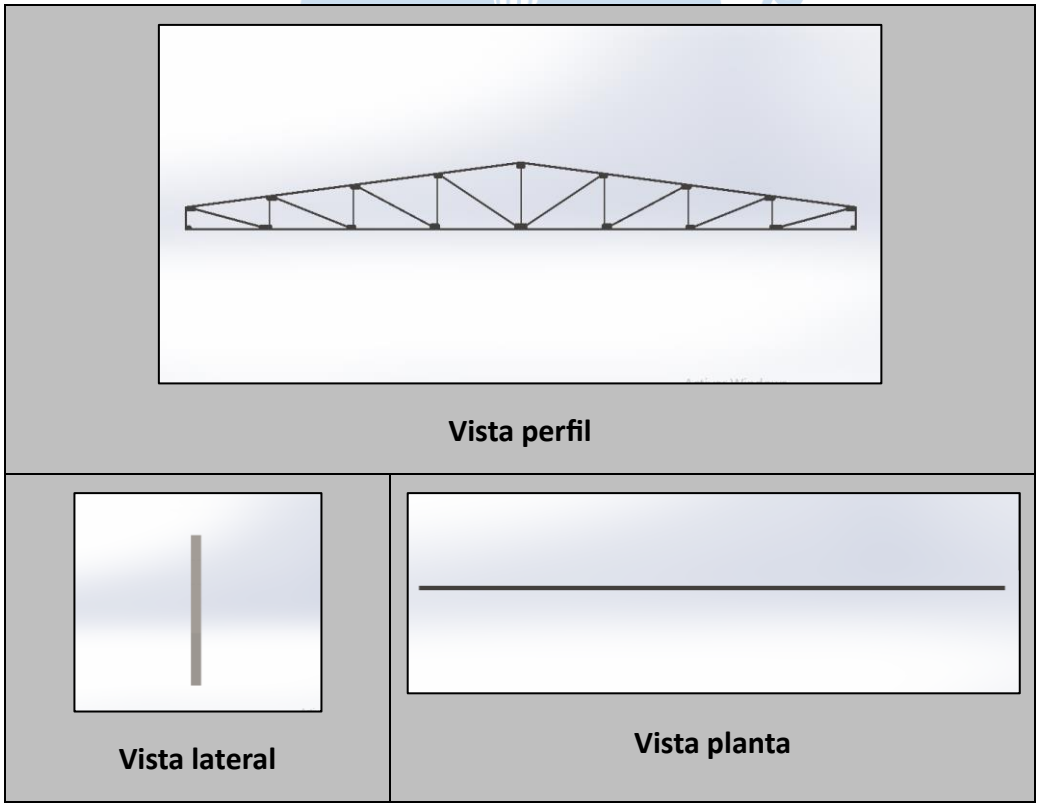
Modelo en 3D del tijeral de 15 metros.



Se muestra una representación visual en la que se observan distintas perspectivas del tijeral, incluyendo vistas de perfil, lateral y planta. (ver figura 42)

Figura 42

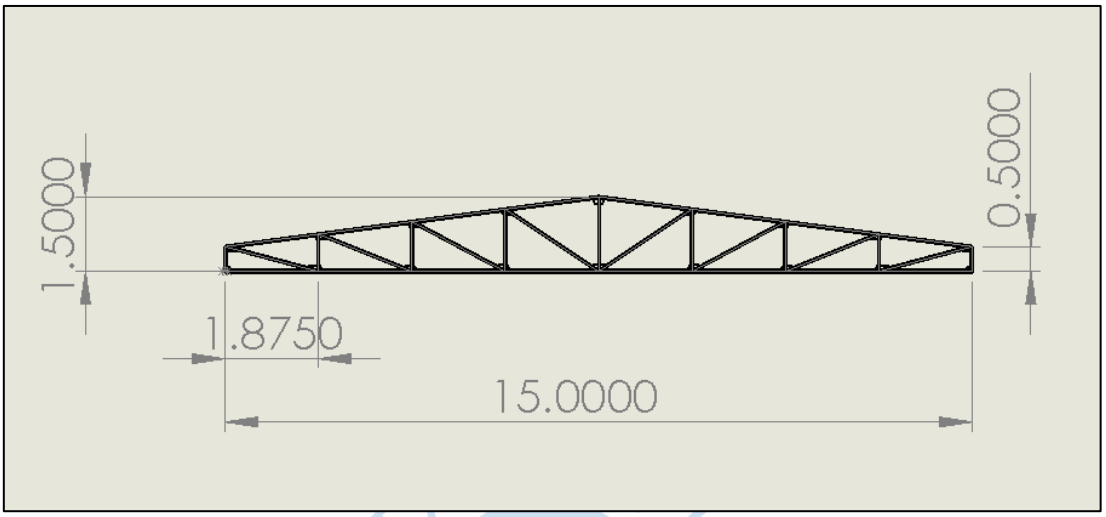
Vistas principales del tijeral de 15 metros.



El diseño y modelado del tijeral se efectuó mediante el software SolidWorks, lo que justifica la presencia de un plano detallado, incluyendo todas las dimensiones necesarias, en la figura 43.

Figura 43

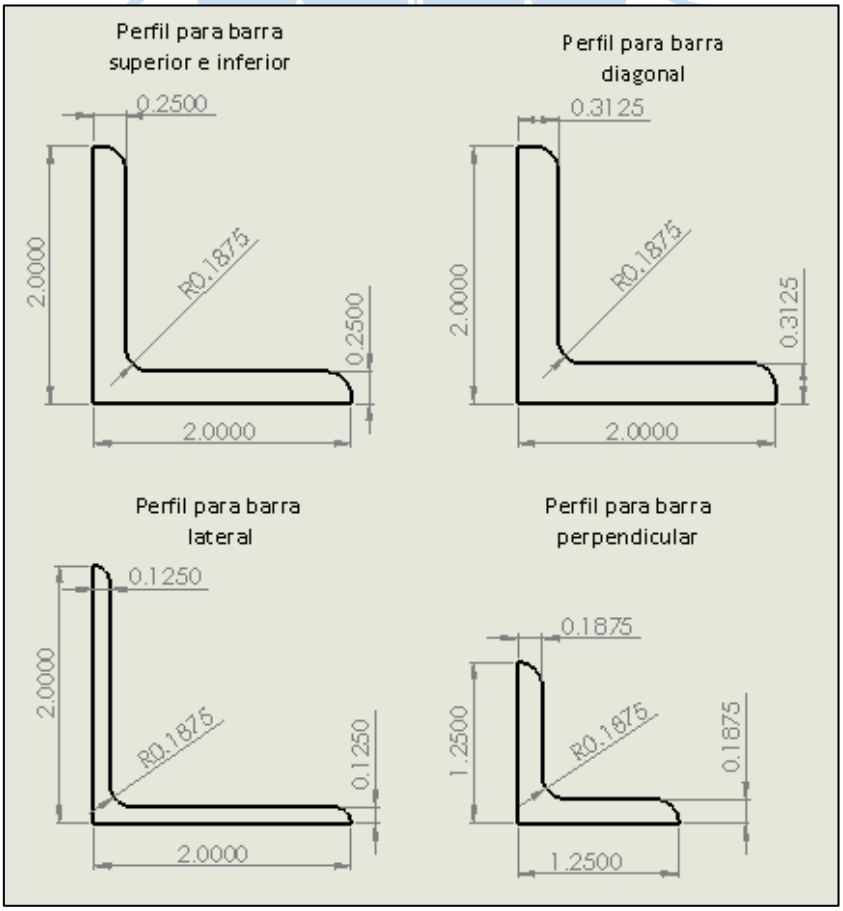
Dimensiones del tijeral de 15 metros.



Es importante destacar que, en la construcción del tijeral de 15 metros, se emplearon perfiles en forma de "L". En la figura 44, se presentan los diversos tipos de perfiles que se utilizaron en la edificación de este tijeral.

Figura 44

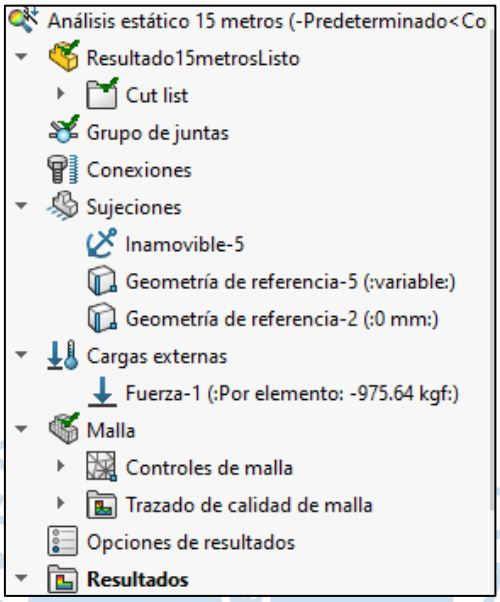
Perfiles en "L" del tijeral de 15 metros.



A continuación, se expone el análisis estático efectuado mediante la herramienta SolidWorks Simulation, donde se detallan los parámetros empleados (ver figura 45)

Figura 45

Parámetros en SolidWorks Simulation del tijeral de 15 metros.



Después de llevar a cabo este análisis en el proyecto, se generan gráficos que muestran las tensiones y desplazamientos resultantes. (ver figura 46 y 47)

Figura 46

Gráfica de tensión del tijeral de 15 metros.

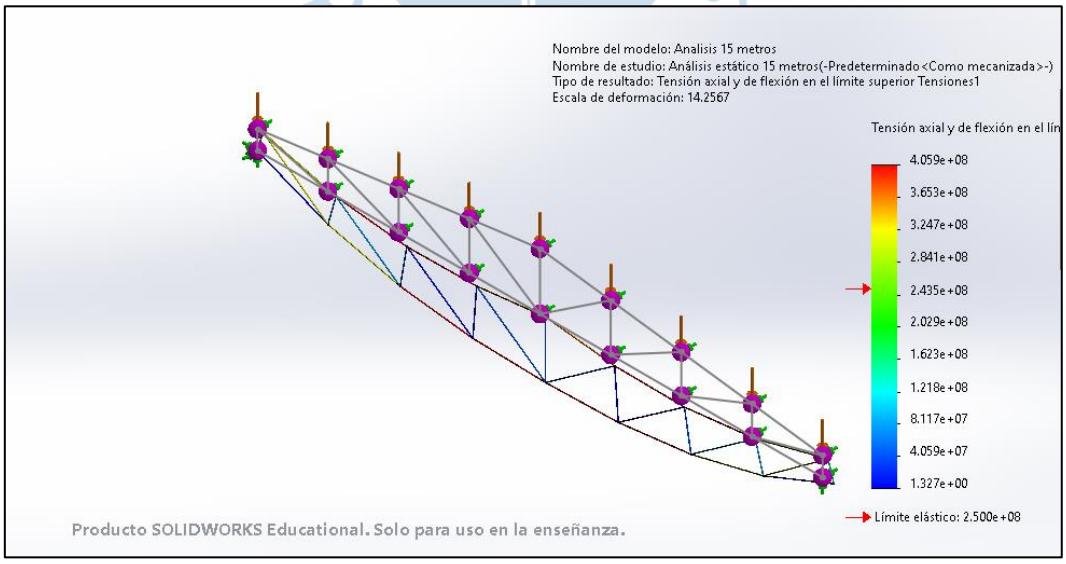
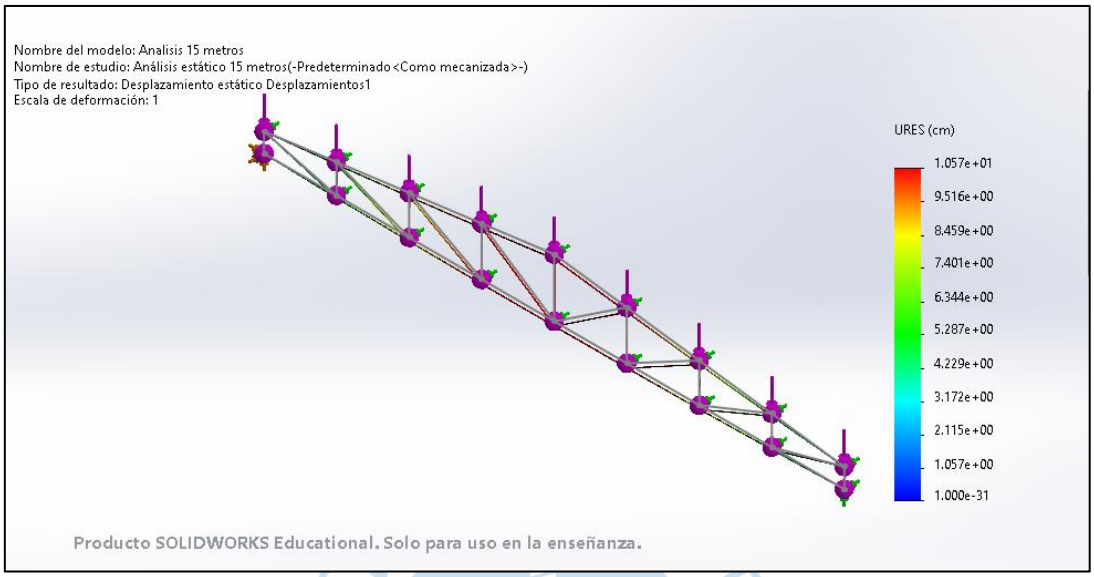


Figura 47

Gráfica de desplazamiento del tijeral de 15 metros.



4.1.2 Resultados obtenidos del caso 1.

Después de llevar a cabo el análisis estático en la estructura, en el cual se consideraron los materiales empleados en el sólido, las restricciones, las cargas aplicadas y la discretización de elementos (mallado), se han obtenido los resultados correspondientes a las fuerzas ejercidas sobra cada barra de la estructura. Estas fuerzas se expresan en kilogramos-fuerza (kgf) y se han evaluado tanto en compresión como en tracción, lo que proporciona una comprensión exhaustiva de las tensiones presentes en el sistema.

Tabla 14

Resultado en SolidWorks Simulation del tijeral de 15 metros.

Nombre de estudio: Análisis estático 15 metros					
Nombre de viga	Axial (kgf)	Nombre de viga	Axial (kgf)	Nombre de viga	Axial (kgf)
Viga-1(Barra 10)	2276 -2276	Viga-12(Barra 16)	2276 -2276	Viga-23(Barra 19)	-2627 2627
Viga-2(Barra 32)	11073 -11073	Viga-13(Barra 5)	-10976 10976	Viga-24(Barra 13)	-1626 1626
Viga-3(Barra 26)	8612 -8612	Viga-14(Barra 29)	9842 -9842	Viga-25(Barra 2)	-8536 8536
Viga-4(Barra 18)	-8835 8835	Viga-15(Barra 1)	0	Viga-26(Barra 15)	975 -975
Viga-5(Barra 12)	0	Viga-16(Barra 24)	-2627 2627	Viga-27(Barra 4)	-10976 10976
Viga-6(Barra 20)	0	Viga-17(Barra 7)	-8536 8536	Viga-28(Barra 28)	11073 -11073
Viga-7(Barra 14)	0	Viga-18(Barra 31)	11073 -11073	Viga-29(Barra 8)	0
Viga-8(Barra 3)	-10976 10976	Viga-19(Barra 11)	975 -975	Viga-30(Barra 23)	0
Viga-9(Barra 17)	4390 -4390	Viga-20(Barra 9)	4390 -4390	Viga-31(Barra 6)	-10976 10976
Viga-10(Barra 22)	1465 -1465	Viga-21(Barra 21)	1465 -1465	Viga-32(Barra 30)	9842 -9842
Viga-11(Barra 27)	11073 -11073	Viga-22(Barra 33)	8612 -8612	Viga-33(Barra 25)	-8835 8835

4.1.3 Comparación de resultados del caso 1

Tabla 15

Comparación de resultados del tijeral de 15 metros

COMPARACION 15 METROS					
FUERZA EN BARRA	MATLAB	SOLIDWORK	FUERZA EN BARRA	MATLAB	SOLIDWORK
	kgf	kgf		kgf	kgf
1	0	0	18	8835	8835
2	8536	8536	19	2626	2627
3	10975	10976	20	0	0
4	10975	10976	21	-1465	-1465
5	10975	10976	22	-1465	-1465
6	10975	10976	23	0	0
7	8536	8536	24	2626	2627
8	0	0	25	8835	8835
9	-4390	-4390	26	-8612	-8612
10	-2276	-2276	27	-11073	-11073
11	-975	-975	28	-11073	-11073
12	0	0	29	-9842	-9842
13	1626	1626	30	-9842	-9842
14	0	0	31	-11073	-11073
15	-975	-975	32	-11073	-11073
16	-2276	-2276	33	-8612	-8612
17	-4390.	-4390			

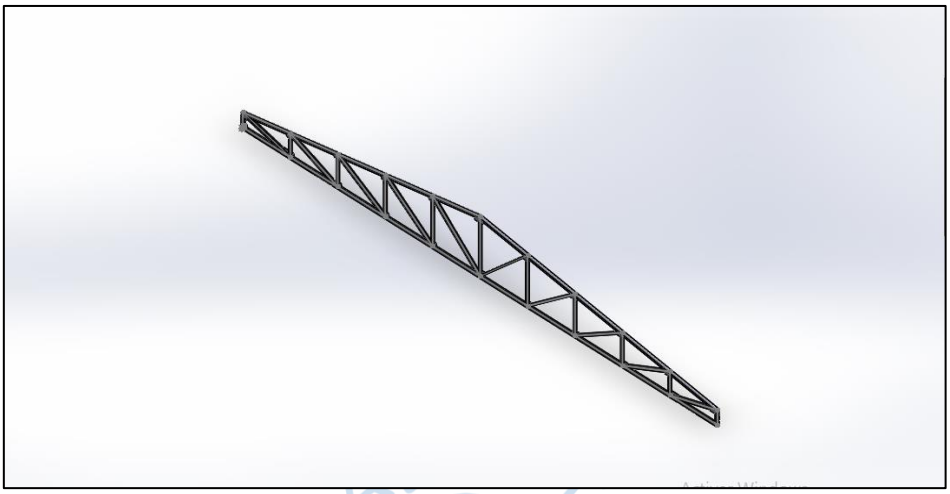
4.2 Caso 2

4.2.1 Diseño del caso 2

En la segunda situación propuesta, se ha establecido que la base del tijeral tenga una dimensión de 20 metros. En consecuencia, en la figura 48 se muestra el modelo tridimensional correspondiente.

Figura 48

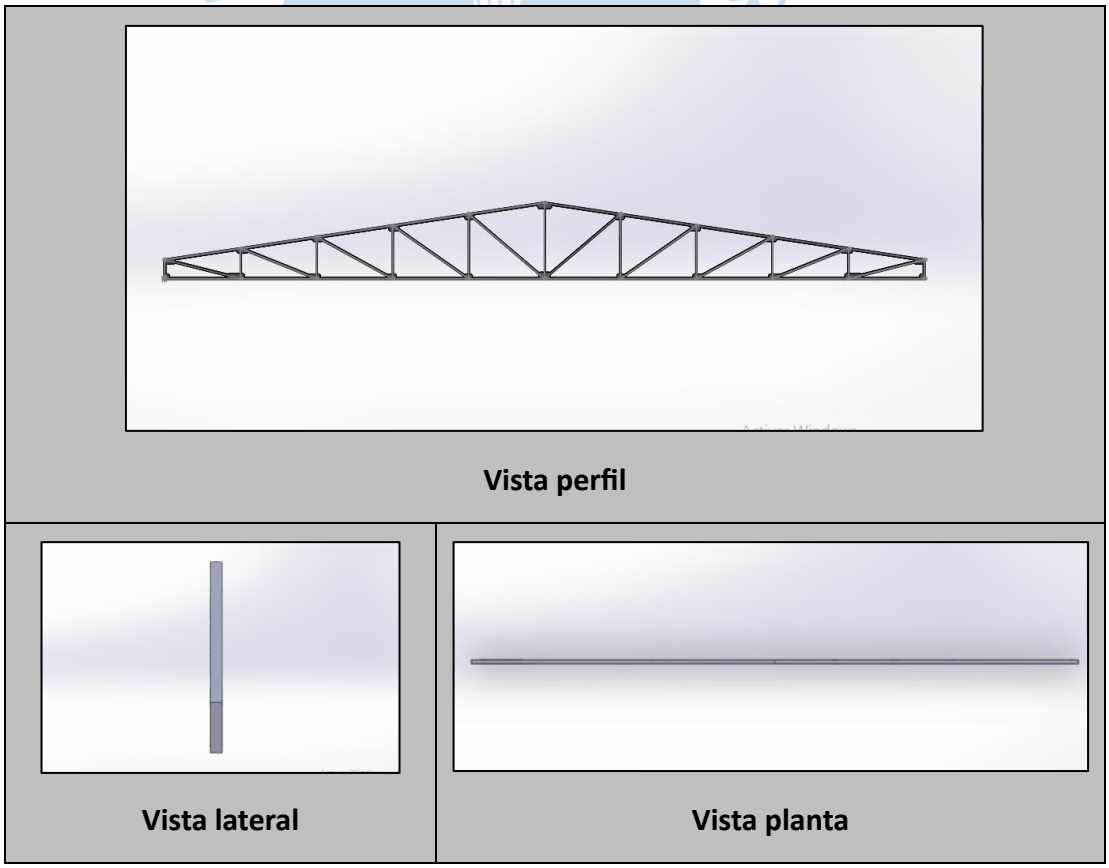
Modelo en 3D del tijeral de 20 metros.



Se muestra una representación gráfica que incluye diversas perspectivas del tijeral, abarcando vistas de perfil, lateral y vista en planta (ver figura 49)

Figura 49

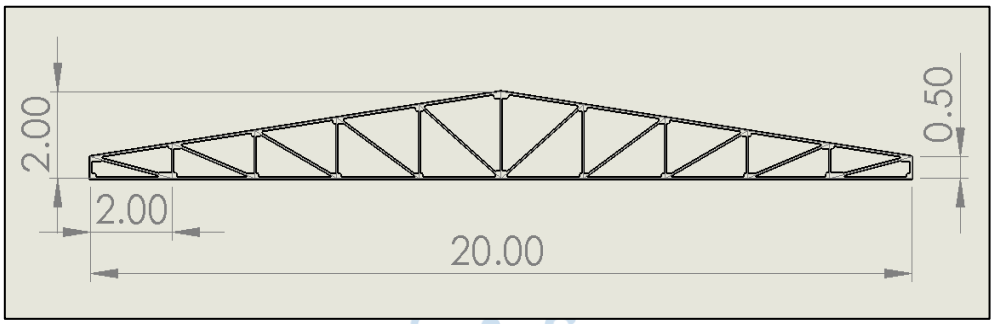
Vistas principales del tijeral de 20 metros.



El tijeral fue diseñado y modelado utilizando el software SolidWorks, lo cual explica la inclusión de un plano detallado que contiene todas las dimensiones requeridas en la figura 50.

Figura 50

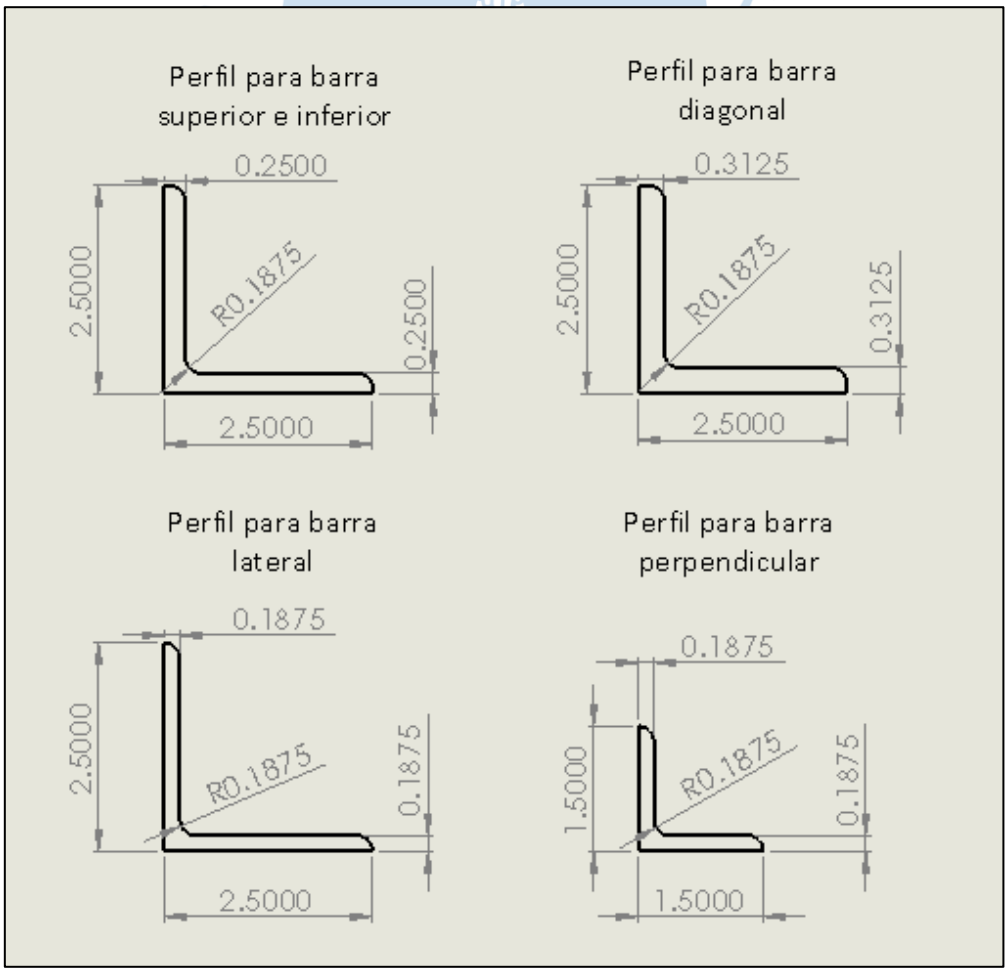
Dimensiones del tijeral de 20 metros.



Es relevante señalar que, durante la construcción del tijeral de 20 metros, se utilizaron perfiles en forma de "L". En la figura 51, se muestran los diferentes tipos de perfiles que se emplearon en la estructura del tijeral.

Figura 51

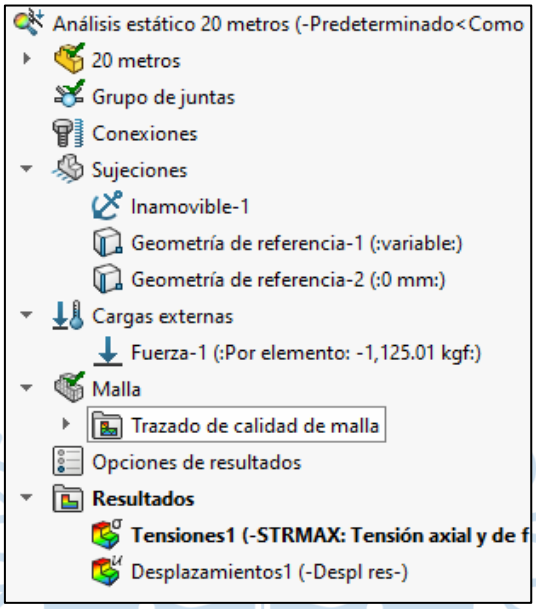
Perfiles en "L" del tijeral de 20 metros.



A continuación, se presenta el análisis estático realizado, utilizando la herramienta SolidWorks Simulation, en el cual se describen en detalle los parámetros utilizados (ver figura 52)

Figura 52

Parámetros en SolidWorks Simulation del tijeral de 20 metros.



Tras completar este análisis en el proyecto, se generan gráficos que representan las tensiones y desplazamientos resultantes (ver figura 53 y 54)

Figura 53

Gráfica de tensión del tijeral de 20 metros.

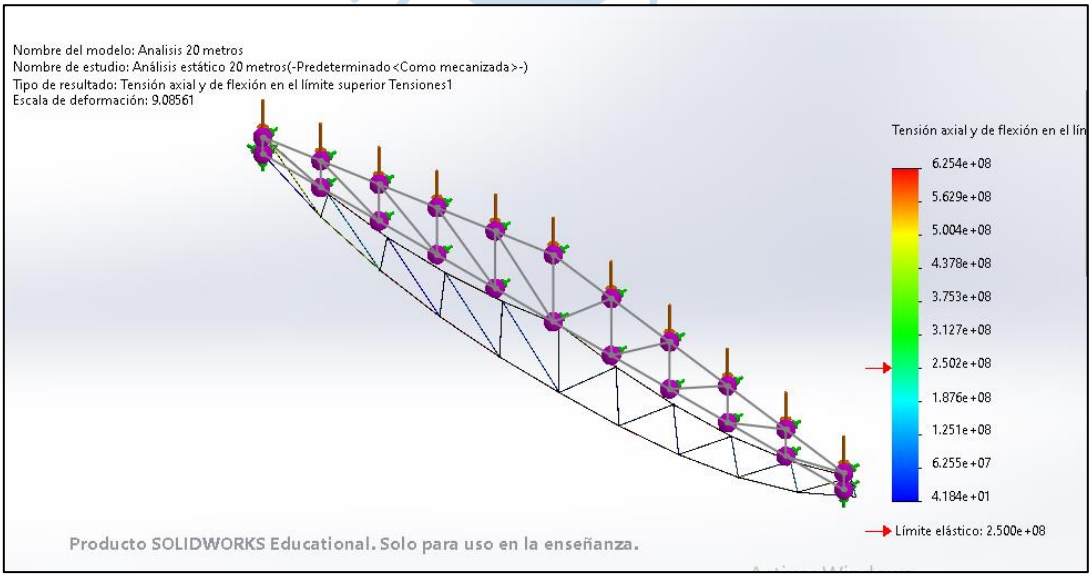
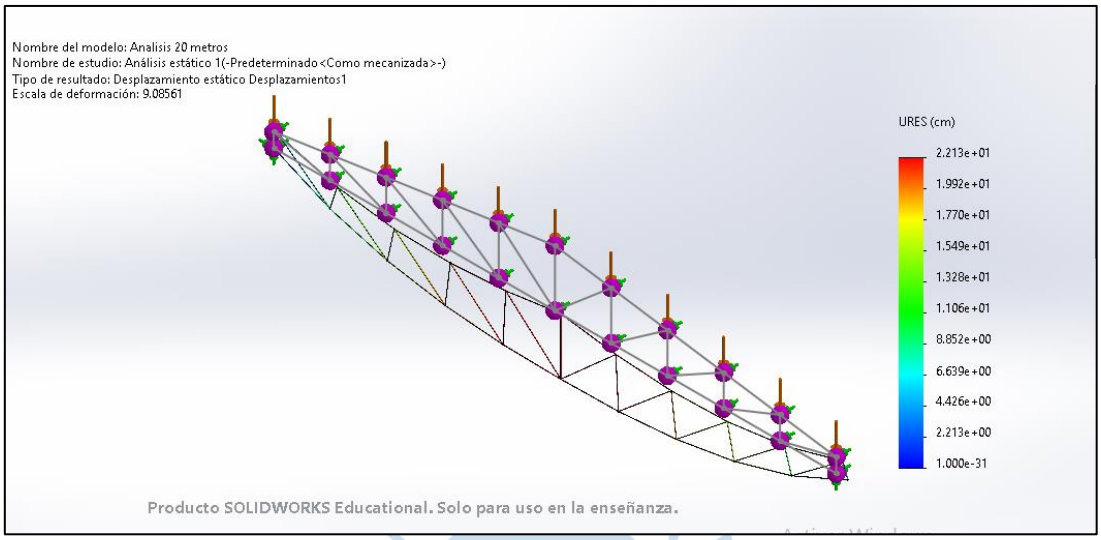


Figura 54

Gráfica de tensión del tijeral de 20 metros.



4.2.2 Resultados obtenidos del caso 2

Tras llevar a cabo un análisis estático detallado, considerando los materiales, restricciones, cargas y mallado en la estructura, se ha obtenido las fuerzas en cada barra. Estas fuerzas, expresadas en kilogramos-fuerza (kgf), tal como se indicó en el caso 1 que estos se han evaluado tanto en compresión como en tracción. Estos resultados son esenciales para garantizar la seguridad y rendimiento de la estructura en el caso 2, con el propósito de poder realizar una comparación con los datos obtenidos en MATLAB.

Tabla 16

Resultado en SolidWorks Simulation del tijeral de 20 metros.

Nombre de estudio: Análisis estático 20 metros					
Nombre de viga	Axial (kgf)	Nombre de viga	Axial (kgf)	Nombre de viga	Axial (kgf)
Viga-1(Barra 23)	-3993 3993	Viga-15(Barra 4)	-16875 16875	Viga-29(Barra 27)	2388 -2388
Viga-2(Barra 40)	16547 -16547	Viga-16(Barra 12)	3164 -3164	Viga-30(Barra 11)	6187 -6187
Viga-3(Barra 19)	1482 -1482	Viga-17(Barra 26)	2388 -2388	Viga-31(Barra 37)	14220 -14220
Viga-4(Barra 24)	-583 583	Viga-18(Barra 30)	-3993 3993	Viga-32(Barra 8)	-16364 16364
Viga-5(Barra 41)	12798 -12798	Viga-19(Barra 34)	17064 -17064	Viga-33(Barra 16)	-3093 3093
Viga-6(Barra 20)	3164 -3164	Viga-20(Barra 5)	-15882 15882	Viga-34(Barra 21)	6187 -6187
Viga-7(Barra 25)	1211 -1211	Viga-21(Barra 13)	1483 -1483	Viga-35(Barra 38)	16060 -16060
Viga-8(Barra 1)	0	Viga-22(Barra 31)	-13046 13046	Viga-36(Barra 9)	-12656 12656
Viga-9(Barra 2)	-12656 12656	Viga-23(Barra 35)	16060 -16060	Viga-37(Barra 17)	-694 694
Viga-10(Barra 28)	1211 -1211	Viga-24(Barra 6)	-15882 15882	Viga-38(Barra 22)	-13046 13046
Viga-11(Barra 32)	12798 -12798	Viga-25(Barra 14)	281 -281	Viga-39(Barra 39)	17064 -17064
Viga-12(Barra 3)	-16364 16364	Viga-26(Barra 36)	14220 -14220	Viga-40(Barra 10)	0
Viga-13(Barra 29)	-583 583	Viga-27(Barra 7)	-16875 16875	Viga-41(Barra 18)	281 -281
Viga-14(Barra 33)	16547 -16547	Viga-28(Barra 15)	-694 694		

4.2.3 Comparación de resultados del caso 2

Tabla 17

Comparación de resultados del tijeral de 20 metros.

COMPARACION 20 METROS					
FUERZA EN BARRA	MATLAB	SOLIDWORKS	FUERZA EN BARRA	MATLAB	SOLIDWORKS
	kgf	kgf		kgf	kgf
1	0	0	22	13045	13046
2	12656	12656	23	3993	3993
3	16363	16364	24	583	583
4	16875	16875	25	-1211	-1211
5	15882	15882	26	-2388	-2388
6	15882	15882	27	-2388	-2388
7	16875	16875	28	-1211	-1211
8	16363	16364	29	583	583
9	12656	12656	30	3993	3993
10	0	0	31	13045	13046
11	-6187	-6187	32	-12797	-12798
12	-3164	-3164	33	-16546	-16547
13	-1482	-1483	34	-17063	-17064
14	-281	-281	35	-16060	-16060
15	694	694	36	-14219	-14220
16	3093	3093	37	-14219	-14220
17	694	694	38	-16060	-16060
18	-281	-281	39	-17063	-17064
19	-1482	-1482	40	-16546	-16547
20	-3164	-3164	41	-12797	-12798
21	-6187	-6187			

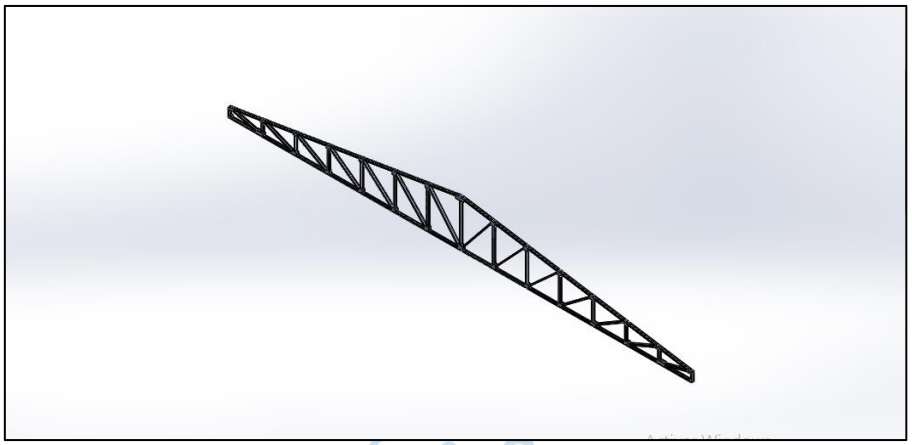
4.3 Caso 3

4.3.1 Diseño del caso 3

En este tercer caso propuesto, se consideró una dimensión de base de 27 metros para el tijeral. Por lo tanto, en la figura 55 se muestra el modelo tridimensional correspondiente.

Figura 55

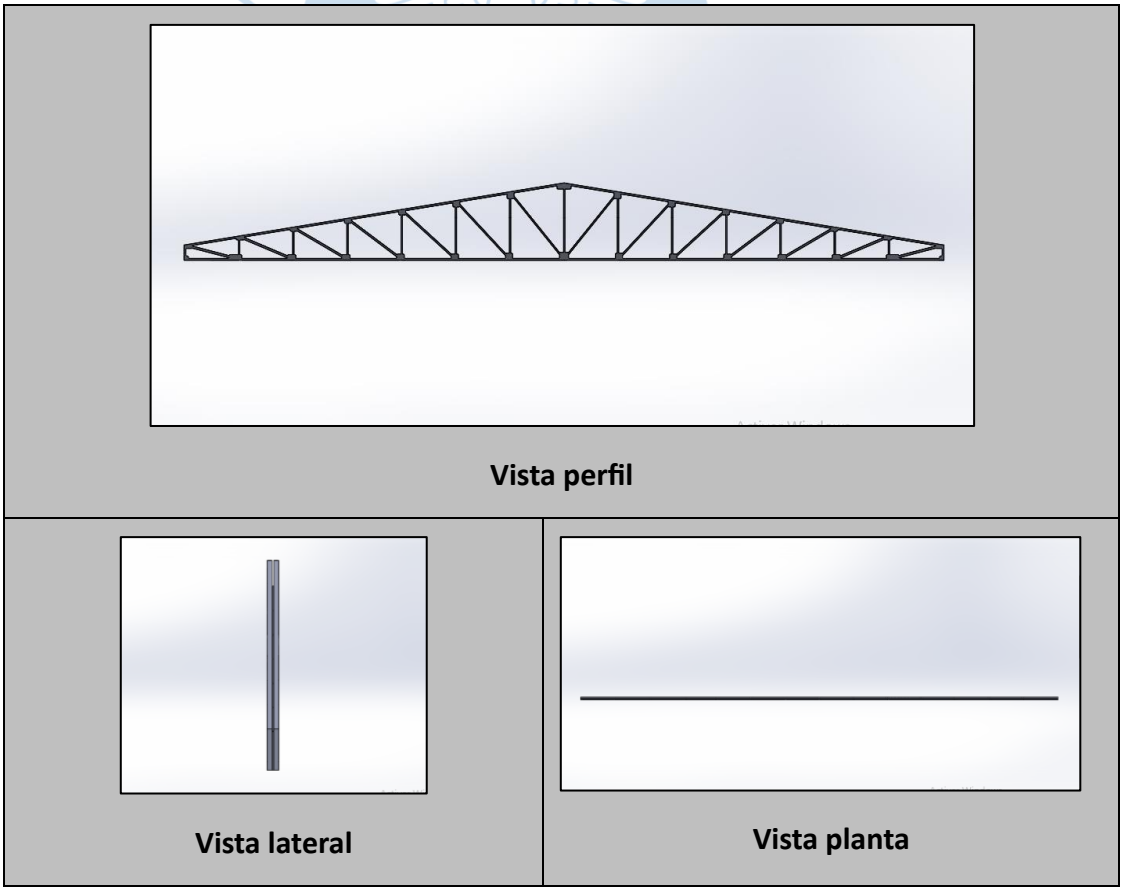
Modelo en 3D del tijeral de 27 metros.



Se presenta una representación gráfica que exhibe diversas perspectivas del tijeral, abarcando vistas desde su perfil, lateral y en planta (ver figura 56)

Figura 56

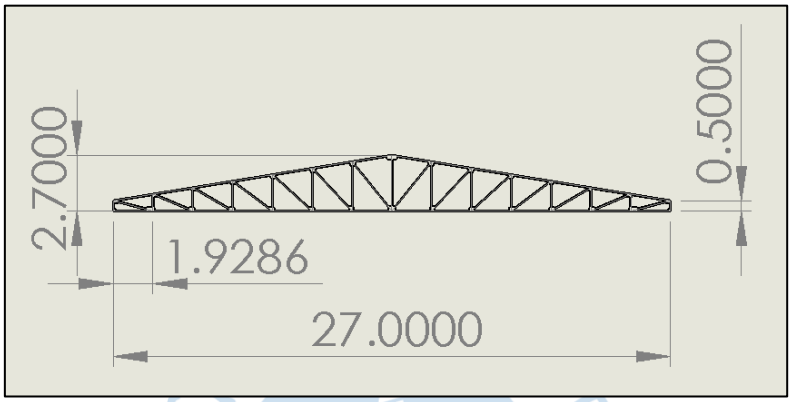
Vistas principales del tijeral de 27 metros.



El diseño y modelado del tijeral se llevaron a cabo mediante el software Solidworks, lo que justifica la presencia de un plano minuciosos que incluye las dimensiones necesarias en la figura 57.

Figura 57

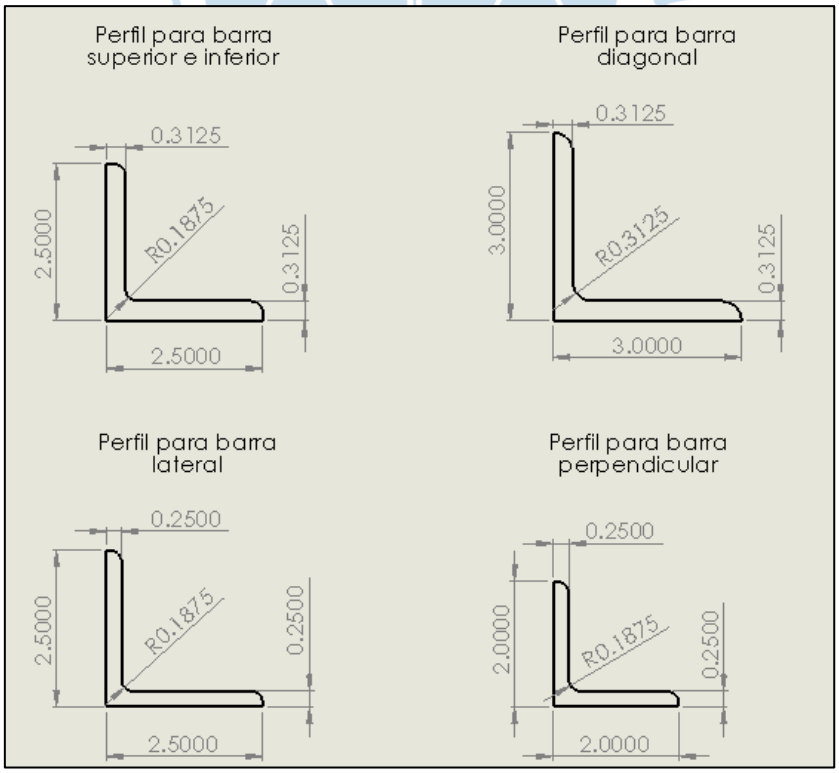
Dimensiones del tijeral de 27 metros.



Es importante destacar que, en la construcción del tijeral de 27 metros, se emplearon perfiles con la forma de "L". En la figura 58, se presentan las distintas variedades de perfiles que se utilizaron en la estructura del tijeral.

Figura 58

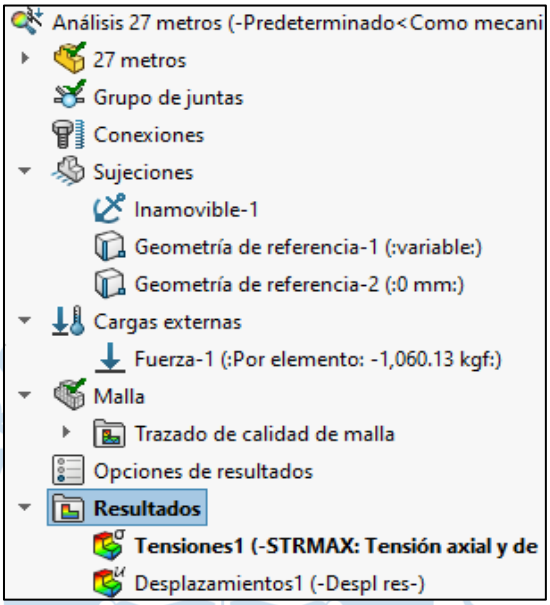
Perfiles en "L" del tijeral de 27 metros.



A continuación, se exhibe un análisis estático realizado mediante la herramienta SolidWorks Simulation, en el que se ofrecen explicaciones detalladas sobre los parámetros empleados (ver figura 59)

Figura 59

Parámetros en SolidWorks Simulation del tijeral de 27 metros.



Luego de realizar el análisis en el proyecto, se generan gráficos que ilustran las tensiones y desplazamientos resultantes (ver figura 60 y 61)

Figura 60

Gráfica de tensión del tijeral de 27 metros.

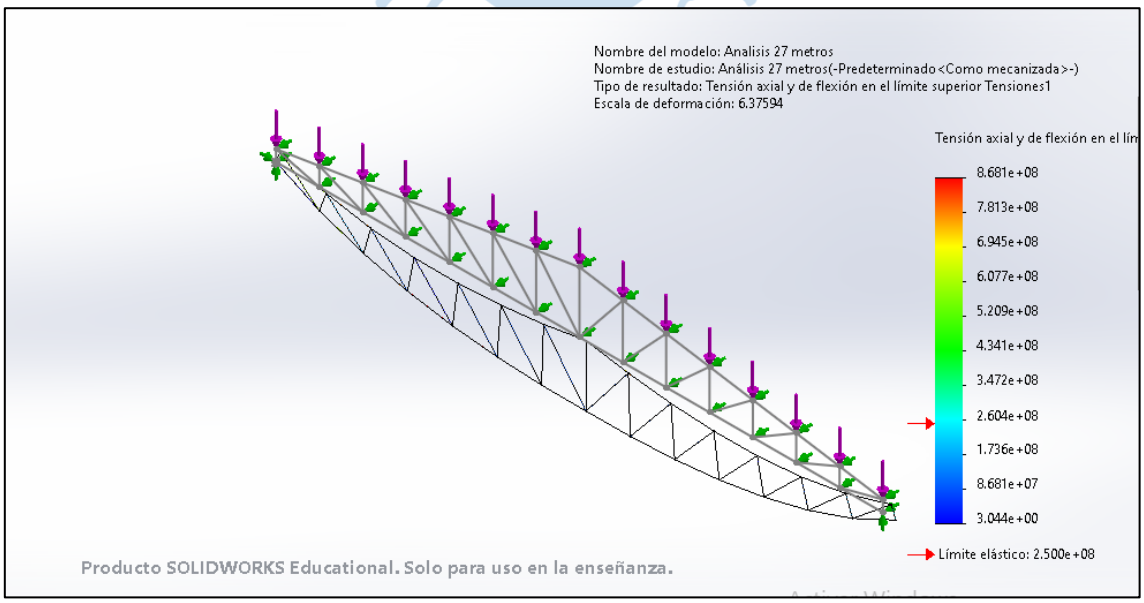
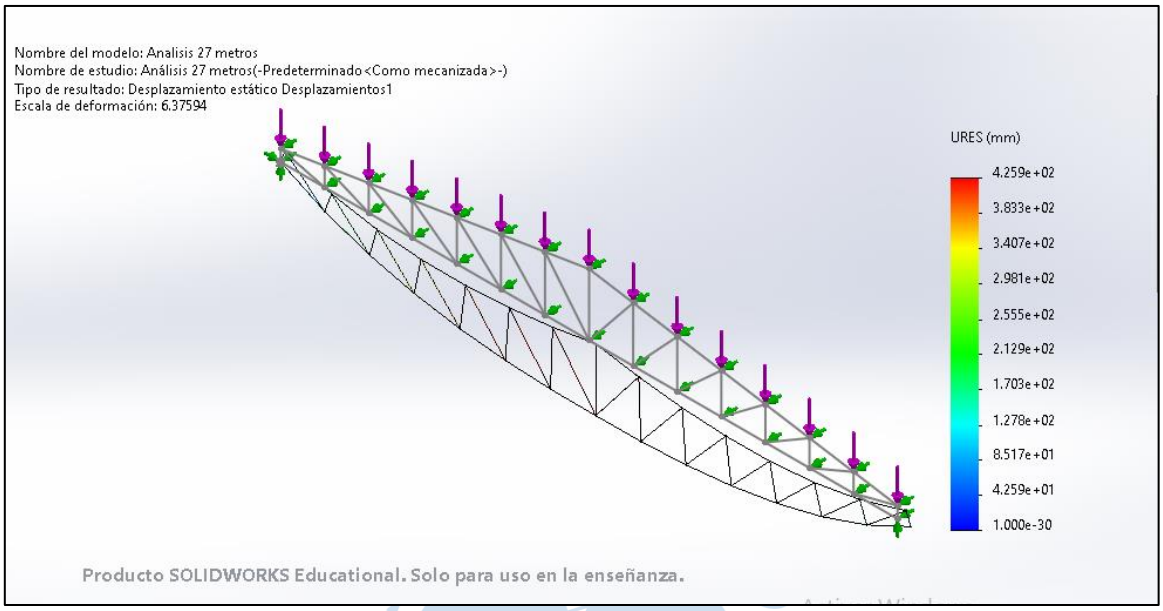


Figura 61

Gráfica de tensión del tijeral de 27 metros.



4.3.2 Resultados obtenidos del caso 3

Luego de un análisis estático detallado de la estructura, considerando material, restricciones, cargas y mallado, se ha obtenido fuerzas en las barras en kgf, evaluadas en compresión y tracción. Estos datos son esenciales para asegurar la seguridad y comparar con resultados de MATLAB, dichos resultados obtenidos en SolidWorks se muestran en la siguiente tabla.

Tabla 18

Resultado en SolidWorks Simulation del tijeral de 27 metros.

Nombre de estudio:Análisis 27 metros					
Nombre de viga	Axial (kgf)	Nombre de viga	Axial (kgf)	Nombre de viga	Axial (kgf)
Viga-1(Barra 9)	-22203 22203	Viga-20(Barra 39)	1438 -1438	Viga-39(Barra 24)	-969 969
Viga-2(Barra 21)	-1759 1759	Viga-21(Barra 6)	-22199 22199	Viga-40(Barra 3)	-21743 21743
Viga-3(Barra 33)	133 -133	Viga-22(Barra 18)	963 -963	Viga-41(Barra 49)	20832 -20832
Viga-4(Barra 55)	23685 -23685	Viga-23(Barra 15)	7952 -7952	Viga-42(Barra 26)	960 -960
Viga-5(Barra 46)	23675 -23675	Viga-24(Barra 52)	20834 -20834	Viga-43(Barra 36)	3203 -3203
Viga-6(Barra 30)	-16863 16863	Viga-25(Barra 41)	-1900 1900	Viga-44(Barra 38)	2407 -2407
Viga-7(Barra 11)	-23376 23376	Viga-26(Barra 8)	-20563 20563	Viga-45(Barra 14)	0
Viga-8(Barra 23)	-1761 1761	Viga-27(Barra 20)	-966 966	Viga-46(Barra 5)	-23260 23260
Viga-9(Barra 35)	2403 -2403	Viga-28(Barra 32)	-1888 1888	Viga-47(Barra 17)	2288 -2288
Viga-10(Barra 2)	-16323 16323	Viga-29(Barra 54)	23573 -23573	Viga-48(Barra 51)	18792 -18792
Viga-11(Barra 37)	3206 -3206	Viga-30(Barra 45)	22030 -22030	Viga-49(Barra 28)	4230 -4230
Viga-12(Barra 57)	16533 -16533	Viga-31(Barra 43)	-16858 16858	Viga-50(Barra 40)	137 -137

Viga-13(Barra 48)	22492 -22492	Viga-32(Barra 10)	-23266 23266	Viga-51(Barra 7)	-20561 20561
Viga-14(Barra 13)	-16318 16318	Viga-33(Barra 22)	-4984 4984	Viga-52(Barra 19)	-78 78
Viga-15(Barra 25)	-82 82	Viga-34(Barra 34)	1434 -1434	Viga-53(Barra 31)	-5883 5883
Viga-16(Barra 4)	-23367 23367	Viga-35(Barra 56)	22022 -22022	Viga-54(Barra 29)	7949 -7949
Viga-17(Barra 16)	4231 -4231	Viga-36(Barra 1)	0	Viga-55(Barra 53)	22496 -22496
Viga-18(Barra 50)	18792 -18792	Viga-37(Barra 47)	23566 -23566	Viga-56(Barra 42)	-5880 5880
Viga-19(Barra 27)	2287 -2287	Viga-38(Barra 12)	-21736 21736	Viga-57(Barra 44)	16538 -16538

4.3.3 Comparación de resultados del caso 3

Tabla 19

Comparación de resultados del tijeral de 27 metros.

COMPARACION 27 METROS					
FUERZA EN BARRA	MATLAB	SOLIDWORKS	FUERZA EN BARRA	MATLAB	SOLIDWORKS
	kgf	kgf		Kgf	kgf
1	0	0	30	16869	16863
2	16329	16323	31	5862	5883
3	21730	21743	32	1910	1888
4	23379	23367	33	-133	-133
5	23272	23260	34	-1433	-1434
6	22212	22199	35	-2417	-2403
7	20565	20561	36	-3201	-3203
8	20565	20563	37	-3201	-3206
9	22212	22203	38	-2417	-2407
10	23272	23266	39	-1433	-1438
11	23379	23376	40	-133	-137
12	21730	21736	41	1910	1900
13	16329	16318	42	5862	5880
14	0	0	43	16869	16858
15	-7950	-7952	44	-16544	-16538
16	-4232	-4231	45	-22018	-22030
17	-2280	-2288	46	-23687	-23675
18	-964	-963	47	-23579	-23566
19	79	78	48	-22505	-22492
20	965	966	49	-20837	-20832
21	1769	1759	50	-18796	-18792
22	4979	4984	51	-18796	-18792
23	1769	1761	52	-20837	-20834
24	965	969	53	-22505	-22496
25	79	82	54	-23579	-23573
26	-964	-960	55	-23687	-23685
27	-2280	-2287	56	-22018	-22022
28	-4232	-4230	57	-16544	-16533
29	-7950	-7949			

Conclusiones

Mediante el uso de dos programas informáticos, uno basado en el lenguaje de programación MATLAB, se llevó a cabo un proceso en el cual se determinaron las coordenadas de los nodos, se evaluaron las fuerzas y se aplicaron restricciones. El propósito inicial fue brindar información sobre qué ángulos en forma de “L” serían apropiados para la construcción del tijeral. Posteriormente, se obtuvieron los valores de las fuerzas en las barras, las cuales se compararon con los resultados obtenidos a través de SolidWorks.

En el proceso de desarrollo utilizando SOLIDWORKS, una herramienta principal para el diseño, modelado y simulación se logró de manera eficiente y accesible la creación de modelos sólidos en 3D. Además, se pudo aprovechar la funcionalidad SOLIWORKS SIMULATION, fundamental para el análisis estructural, que permitió aplicar cargas, restricciones en los nodos seleccionados con el objetivo de calcular las fuerzas en las barras. Este enfoque integral de trabajo con ambos programas SOLIDWORKS y MATLAB, ha sido el núcleo de este proyecto, con el propósito de llevar a cabo una comparación exhaustiva de los resultados obtenidos.

Estos modelos computacionales, implementados en SOLIDWORKS y Matlab, se presentan como recursos fundamentales tanto para los profesionales enfocados en el análisis estructural como para los estudiantes inmersos en cursos de esta disciplina. Les proporcionan herramientas de confianza que simplifican la comparación y validación de sus cálculos, garantizando la precisión y confiabilidad en los distintos análisis estructurales.

Recomendaciones

Los profesionales con experiencia en programación tienen el potencial de identificar formas más simplificadas de desarrollar este software, lo que podría convertirlo en un modelo comercial en el futuro. No obstante, el objetivo principal de este modelo es fomentar que cualquier estudiante pueda comprender, analizar y contribuir a futuras mejoras en el programa.

El presente código estará disponible para cualquier tesista interesado en investigar la programación de distintos tipos de diagramas. Se recomienda ponerlo a disposición de futuras investigaciones, permitiendo la utilización como base para ampliar la funcionalidad en el ámbito del diseño estructural. Esto fomentará oportunidades para que otros investigadores puedan explorar y utilizar el código en futuras investigaciones.

Referencias

Afrazi, M., Pirjalili, A., y Golshani, A. (2017, septiembre). *MATLAB codes for Finite Element Analysis of a Truss.* [ponencia]. 4th International Conference on Recent Innovations in Civil Engineering, Architecture an Urban Planning, Irán. https://www.researchgate.net/publication/341576520_MATLAB_codes_for_Finite_Element_Analysis_of_a_Truss

Bakošová, A., Krmela, J., y Handrik, M. (2020, agosto). Computing of Truss Structure using MATLAB. *MANUFACTURING TECHNOLOGY, 20*(3), 279-285. https://www.journalmt.com/pdfs/mft/2020/03/09.pdf

Chachalo, M. (2023). *Diseño de una estructura metálica con un área de 400 m2 para una línea de galvanizado en caliente.* [tesis de grado, Universidad Politécnica Salesiana]. Repositorio Institucional UPS. http://dspace.ups.edu.ec/handle/123456789/24578

Coronado, A., & Vásqiez, C. (2023). *Modelado y construcción a escala de una fresadora convencional impresa en 3D para fines académicos*. [tesis de grado, Universidad de Piura]. Repositorio Institucional UDEP. https://hdl.handle.net/11042/6029

Cruz, E., & Hancco, J. (2023). *Diseño estructural en acero del edificio comercial Manuel Prado utilizando el CYPE 3D.* [tesis de grado, Universidad Nacional de San Antonio Abad del Cusco]. Repositorio Institucional UNSAAC. https://repositorio.unsaac.edu.pe/handle/20.500.12918/7422

Diaz, F. (2020). *Modelo computacional para el análisis matricial de estructuras reticulares.* [tesis de grado, Universidad Peruana de Ciencias Aplicadas]. Repositorio Institucional UPC. http://hdl.handle.net/10757/648845

Fabró, A. (2020). *Programación de un código de elementos finitos para el cálculo de estructuras.* [tesis de grado, Universidad Politécnica de Cataluña, España]. Repositorio Institucional UPCommons. https://upcommons.upc.edu/handle/2117/328642

Faragó, R., Sousa, M. y Riqueira, A. (2021) Resistance factor calibration for perforated cold-formed steel compression members. *RIC, 37*(1), 35-46. http://dx.doi.org/10.7764/RIC.00015.21

González, O., González C. , y López, A. (2020). Introducción al método del elemento finito: Solidworks y Matlab. *Ideas en Ciencias de la Ingeniería, 1*(1), 27-47. https://ideasencienciasingenieria.uaemex.mx/article/view/14589

Goñi, D., y Cáceres, J. (2018). *Comparativo técnico-económico de una nave industrial con un sistema de tijerales y de pórticos.* [tesis de grado, Pontificia Universidad Católica del Perú]. Repositorio Institucional PUCP. http://hdl.handle.net/20.500.12404/10193

Huallpa, R. (2017). *Diseño y verificación estructural de un soporte para una faja transportadora de capacidad de 120TM a 140TM de mineral de cobre.* [tesis de grado, Universidad Nacional de San Agustín de Arequipa]. Repositorio Institucional UNAS. http://repositorio.unsa.edu.pe/handle/UNSA/6134

Maza, Victor. (2021). *Modelado y construcción a escala de torno convencional impreso en 3D para fines académicos.* [tesis de grado, Universidad de Piura]. Repositorio Institucional UDEP. https://hdl.handle.net/11042/5121

Morató, J. (2015). *Desarrollo en Matlab de software de cálculo de placas por MEF.* [tesina de especialidad, Universidad Politécnica de Cataluña, España.] Repositorio Institucional UPCommons. https://upcommons.upc.edu/handle/2117/78420?show=full

Ramos, R. (2022). Diseño estructural en acero para el mejoramiento de los servicios de educación seundaria en la I.E. Champagnat en el distrito de Tacna, provincia de Tacna- Tacna - Perú. [tesis de grado, Universidad Nacional Experimental del Táchira]. Repositorio Institucional RENATI. https://renati.sunedu.gob.pe/handle/sunedu/3410778

Segui, W. (2000). *Diseño de estructuras de acero con LRFD* (2.ª ed). México: International Thomson.

Torres, O. y Aroca, A. (2015). *Desarrollo de una aplicación de software en MATLAB para el cálculo de esfuerzos y deformaciones planos utilizando el método de elementos finitos.* [tesis de grado, Universidad Distrital Francisco José de Caldas]. Repositorio Institucional UDISTRITAL. http://hdl.handle.net/11349/3652

Villar, C. (2017). *Extensión para el modelamiento, análisis y diseño automatizado de techos retráctiles en SAP 2000 con paneles solares integrados a la cubierta.* [tesis de maestría, Universidad Nacional de Colombia]. Repositorio Institucional UN. https://repositorio.unal.edu.co/handle/unal/63968

Zapata, L. (1997). *Diseño estructural en acero* (2.ª ed). Lima: Luis F. Zapata Baglietto.

Apéndices

Apéndice A Código de programación – Tijeral

```
clear all
clc
% Datos con los que se partirá para los cálculos del tijeral
% h es la altura lateral del tijeral
h=0.5;
% L es la longitud de frontera del tijeral
L=input('Ingresa el valor de L en metros = ');
% H es la altura máxima que tomará el tijeral
H= L/10;
% L_1 es la pendiente donde irán las calaminas o planchas
L_1= sqrt((H-h)^2+(L/2)^2);
L_1=ceil(L_1*100)/100;
% Calcula el ángulo en radianes
angulo_radianes = atan((H - h) / (L / 2));
% Convierte el ángulo a grados
angulo_grados = rad2deg(angulo_radianes);

% Selección de planchas dadas sus medias en metros:
%Plancha de 6 pies
P_1= 1.8;
%Plancha de 7 pies
P_2= 2.1;
%Plancha de 8 pies
P_3= 2.4;
% Selección del traslape o junta de calamina, se recomienda 0.15m a 0.2m
t=input('Ingresa el valor de t en metros = ');
P_1=P_1-t;
P_2=P_2-t;
P_3=P_3-t;

% Selección de la cantidad de calaminas a usar en la longitud de la pendiente:
syms N_1 N_2 N_3; % Definir variables simbólicas de números de calaminas que
se tomaran por cada longitud de plancha
numero = L_1; % Número a dividir
divisores = [P_1,P_2,P_3]; % Divisores a utilizar
resultados = []; % Almacenar los resultados de la división
% Cantidad de calaminas a usar
for i = 1:length(divisores)
divisor = divisores(i);
resultado = ceil(numero / divisor);
fprintf(['El resultado de dividir la %.2f entre la plancha P_%d resulta el
numero de calaminas N_%d igual a %.f\n'],L_1,i,i, resultado);
resultados = [resultados, resultado]; % Almacenar el resultado en el vector
de resultados
end
% Convertir los resultados en valores numéricos y almacenarlos en el workspace
N_1 = double(resultados(1));
N_2 = double(resultados(2));
N_3 = double(resultados(3));
fprintf('N_1: %.f\n', N_1);
fprintf('N_2: %.f\n', N_2);
fprintf('N_3: %.f\n', N_3);

syms Lf_1 Lf_2 Lf_3; % Definir variables simbólicas de la longitud final
obtenida por cada plancha
producto = 1; % Inicializar el producto
```

```
productos = [];
for i = 1:length(resultados)
producto = divisores(i) * resultados(i);
fprintf('El valor de multiplicar N_%d por P_%d nos da como resultado la
longitud final Lf_%d igual a %.2f\n', i, i, i,producto);
productos =[productos, producto]; % Almacenar el producto en el vector de
productos
end
% Convertir los resultados en valores numéricos y almacenarlos en el workspace
Lf_1 = double(productos(1));
Lf_2 = double(productos(2));
Lf_3 = double(productos(3));
fprintf('Lf_1: %.2f\n', Lf_1);
fprintf('Lf_2: %.2f\n', Lf_2);
fprintf('Lf_3: %.2f\n', Lf_3);

%Obtener la longitud de la pendiente verdadera tabulada con las obtenidas
fianles como Lf_i
valores = [Lf_1, Lf_2, Lf_3]; % Lista de valores
objetivo = L_1; % Valor objetivo
valor_cercano = Inf; % Valor inicialmente infinito
diferencia_minima = 0.2; % Diferencia mínima requerida
for i = 1:length(valores)
valor_actual = valores(i);
if valor_actual >= objetivo && valor_actual < valor_cercano && (valor_actual
- objetivo) >= diferencia_minima
valor_cercano = valor_actual;
end
end
fprintf('El valor más cercano y mayor a %.2f es %.2f\n', objetivo,
valor_cercano);

% Determinar el índice de la calamina utilizada
indice = find(valores == valor_cercano);
% Determinar el tipo de calamina correspondiente al índice
tipo_calamina = "";
switch indice
case 1
tipo_calamina = 'P_1';
case 2
tipo_calamina = 'P_2';
case 3
tipo_calamina = 'P_3';
end
fprintf('Por lo tanto, se utilizará la calamina %s\n', tipo_calamina);
fprintf('El número de calaminas y espacios por lado a usar será %d\n',
resultado);
% Coordenadas de los nodos [x y]
H1 = h; % altura de la parte rectangular
H2 = H-h; % altura de la parte triangular
n = 2*resultado; % número de divisiones en la base
% Calcular el espaciado entre nodos en la parte de abajo
dL = L / n;
dH = H1 / n;
% Calcular las coordenadas de los nodos en la parte rectangular
nodos = [];
for j = 0:n
x = j * dL;
```

```
y = 0;
nodos = [nodos; x, y];
end
% Calcular la cantidad de divisiones en cada recta de la parte triangular
n_divisiones = n / 2;
% Calcular las coordenadas de los nodos en la parte triangular
for i = 0:n_divisiones-1
% Calcular las coordenadas x de los nodos en los bordes izquierdo y derecho
x_left = (i / n_divisiones) * (L / 2);
x_right = L - x_left;
% Calcular la coordenada y de los nodos en el borde superior
y = H1 + (i / n_divisiones) * H2;
% Agregar los nodos al conjunto de nodos
nodos = [nodos; x_left, y];
nodos = [nodos; x_right, y];
end
% Agregar el nodo en la punta superior
nodos = [nodos; L / 2, H1 + H2];
% Graficar los nodos
figure; % Crear una nueva figura
plot(nodos(:, 1), nodos(:, 2), 'rO'); % Graficar los nodos como puntos rojos
hold on;
% Enumerar los nodos en un solo orden y sentido
num_nodos = size(nodos, 1);
offset = 0.2; % Desplazamiento lateral para los números de nodo
barra_num = 1; % Variable para enumerar las barras
% Obtener la cantidad de nodos en la matriz nodos
num_nodos = size(nodos, 1);
%Calcular la fuerza que será distribuido en cada nodo para eso se debe
ingresar el valor del largo de las divisiones establecidos a lo largo del
área que se tiene
largo_division=input('Ingresa el valor de largo_division en metros = ');
Area_pendiente= valor_cercano*largo_division/n_divisiones;
Peso_calamina= 10; % en kg/m^2
Peso_estructura=15; % en kg/m^2
Carga_viva_techo=30;% en kg/m^2
Presion_dinamica= 0.005*(55^2); % en kg/m^2
Carga_viento=-1.2*Presion_dinamica*Area_pendiente; % en kg
Carga_viva_nodo=-Area_pendiente*Carga_viva_techo; % en kg
Carga_muerta_nodo= -Area_pendiente*(Peso_calamina+Peso_estructura); % en kg
Carga_total_nodo=     1*Carga_viva_nodo     +     1.2*Carga_muerta_nodo     +
1.3*Carga_viento*cos(angulo_radianes); % en kg

% Generar los nodos de fuerza desde n+2 hasta 2n+2
nodos_fuerza = (n+2):(2*n+2);
% Asignar las fuerzas en X y Y para los nodos de fuerza
fuerza_X = zeros(length(nodos_fuerza), 1);
fuerza_Y = Carga_total_nodo * ones(length(nodos_fuerza), 1);

% Validar la longitud de los vectores de fuerza
if   length(fuerza_X)   ~=   length(nodos_fuerza)   ||   length(fuerza_Y)   ~=
length(nodos_fuerza)
error('La longitud de los vectores de fuerza no coincide con la cantidad de
nodos especificada.');
end

% Graficar y unir nodos
for i = 1:2*n +2
```

```
% Obtener las coordenadas del nodo actual
x = nodos(i, 1);
y = nodos(i, 2);
% Graficar el número del nodo
text(x + offset, y, num2str(i), 'Color', 'black', 'HorizontalAlignment',
'left', 'VerticalAlignment', 'middle');
% Unir el nodo actual con el siguiente nodo para las barras inferiores
if i < n+1
x_next = nodos(i+1, 1);
y_next = nodos(i+1, 2);
line([x, x_next], [y, y_next], 'Color', 'blue');
% Enumerar la barra entre el nodo actual y el siguiente nodo
text((x + x_next) / 2, (y + y_next) / 2, num2str(barra_num), 'Color', 'red',
'HorizontalAlignment', 'center', 'VerticalAlignment', 'middle');
barra_num = barra_num + 1;
end
end

% Unir los nodos de la parte izquierda inclinada del tijeral
for i = n+2:2:2*n
% Obtener las coordenadas de los nodos
x1 = nodos(i, 1);
y1 = nodos(i, 2);
x2 = nodos(i+2, 1);
y2 = nodos(i+2, 2);
% Unir los nodos con una línea
plot([x1, x2], [y1, y2], 'b-', 'LineWidth', 1);
% Calcular las coordenadas del punto medio entre los nodos
x_mid = (x1 + x2) / 2;
y_mid = (y1 + y2) / 2;
% Enumerar la barra
num_barra = (i - n - 2) / 2 + 3*n+2;
text(x_mid,      y_mid,      num2str(num_barra),      'Color',      'red',
'HorizontalAlignment', 'center', 'VerticalAlignment', 'middle');
end
% Unir el nodo 2n+2 con el nodo 2n+1
plot([nodos(2*n+2, 1), nodos(2*n+1, 1)], [nodos(2*n+2, 2), nodos(2*n+1, 2)],
'b-', 'LineWidth', 1);
% Obtener las coordenadas de los nodos
x1 = nodos(2*n+2, 1);
y1 = nodos(2*n+2, 2);
x2 = nodos(2*n+1, 1);
y2 = nodos(2*n+1, 2);
% Unir los nodos con una línea
plot([x1, x2], [y1, y2], 'b-', 'LineWidth', 1);
% Calcular las coordenadas del punto medio entre los nodos
x_mid = (x1 + x2) / 2;
y_mid = (y1 + y2) / 2;
% Enumerar la barra
num_barra = 3*n+n_divisiones + 2;
text(x_mid,      y_mid,      num2str(num_barra),      'Color',      'red',
'HorizontalAlignment', 'center', 'VerticalAlignment', 'middle');
% Unir los nodos de la parte derecha inclinada del tijeral
for i = 2*n+1:-2:n+5


% Obtener las coordenadas de los nodos
x1 = nodos(i, 1);
```

```
y1 = nodos(i, 2);
x2 = nodos(i-2, 1);
y2 = nodos(i-2, 2);
% Unir los nodos con una línea
plot([x1, x2], [y1, y2], 'b-', 'LineWidth', 1);
% Calcular las coordenadas del punto medio entre los nodos
x_mid = (x1 + x2) / 2;
y_mid = (y1 + y2) / 2;
% Enumerar la barra
if n_divisiones == 3
num_barra = 4 * n + n_divisiones - ceil((i - n - 1) / 2);
elseif n_divisiones == 4
num_barra = 4 * n + n_divisiones - (i - n + 1) / 2;
elseif n_divisiones == 5
num_barra = 4 * n + n_divisiones - 1 - (i - n + 1) / 2;
elseif n_divisiones == 6
num_barra = 4 * n + n_divisiones-2 - (i - n + 1) / 2;
elseif n_divisiones == 7
num_barra = 4 * n + n_divisiones-3 - (i - n + 1) / 2;
end
text(x_mid,       y_mid,       num2str(num_barra),       'Color',       'red',
'HorizontalAlignment', 'center', 'VerticalAlignment', 'middle');
end

% Unir los nodos 1 con 10, 2 con 12, 3 con 14, 4 con 16, 5 con 18, etc.
for i = 1:n_divisiones+1
% Obtener los índices de los nodos
idx1 = i;
idx2 = 2 * i + n;
% Obtener las coordenadas de los nodos
x1 = nodos(idx1, 1);
y1 = nodos(idx1, 2);
x2 = nodos(idx2, 1);
y2 = nodos(idx2, 2);
% Unir los nodos con una línea
plot([x1, x2], [y1, y2], 'b-', 'LineWidth', 1);
% Calcular las coordenadas del punto medio entre los nodos
x_mid = (x1 + x2) / 2;
y_mid = (y1 + y2) / 2;
% Enumerar la barra
num_barra = i + n;
text(x_mid,       y_mid,       num2str(num_barra),       'Color',       'red',
'HorizontalAlignment', 'center', 'VerticalAlignment', 'middle');
end
% Unir los nodos 6 con 17, 7 con 15, 8 con 13, etc.
for i = 1:n_divisiones

% Obtener los índices de los nodos
idx1 = i + n_divisiones + 1;
idx2 = 2*n+3 - (2 * i);
% Obtener las coordenadas de los nodos
x1 = nodos(idx1, 1);
y1 = nodos(idx1, 2);
x2 = nodos(idx2, 1);
y2 = nodos(idx2, 2);

% Unir los nodos con una línea
plot([x1, x2], [y1, y2], 'b-', 'LineWidth', 1);
```

```
% Calcular las coordenadas del punto medio entre los nodos
x_mid = (x1 + x2) / 2;
y_mid = (y1 + y2) / 2;
% Enumerar la barra
num_barra = i + n+n_divisiones+1;
text(x_mid,       y_mid,       num2str(num_barra),       'Color',       'red',
'HorizontalAlignment', 'center', 'VerticalAlignment', 'middle');
end

% Unir los nodos 2 con 10, 3 con 12, 4 con 16, 5 con 16, etc.
for i = 1:n_divisiones
% Obtener los índices de los nodos
idx1 = i + 1;
idx2 = n + (2 * i);
% Obtener las coordenadas de los nodos
x1 = nodos(idx1, 1);
y1 = nodos(idx1, 2);
x2 = nodos(idx2, 1);
y2 = nodos(idx2, 2);
% Unir los nodos con una línea
plot([x1, x2], [y1, y2], 'b-', 'LineWidth', 1);
% Calcular las coordenadas del punto medio entre los nodos
x_mid = (x1 + x2) / 2;
y_mid = (y1 + y2) / 2;

% Enumerar la barra
num_barra = i+2*n+1;
text(x_mid,       y_mid,       num2str(num_barra),       'Color',       'red',
'HorizontalAlignment', 'center', 'VerticalAlignment', 'middle');
end
% Unir los nodos 5 con 17, 6 con 15, 7 con 13, 8 con 11, etc.
for i = 1:n_divisiones
% Obtener los índices de los nodos
idx1 = i + n_divisiones;
idx2 = 2*n+3 - (2 * i);
% Obtener las coordenadas de los nodos
x1 = nodos(idx1, 1);
y1 = nodos(idx1, 2);
x2 = nodos(idx2, 1);
y2 = nodos(idx2, 2);
% Unir los nodos con una línea
plot([x1, x2], [y1, y2], 'b-', 'LineWidth', 1);
% Calcular las coordenadas del punto medio entre los nodos
x_mid = (x1 + x2) / 2;
y_mid = (y1 + y2) / 2;

% Enumerar la barra
num_barra = i+2*n+n_divisiones+1;
text(x_mid,       y_mid,       num2str(num_barra),       'Color',       'red',
'HorizontalAlignment', 'center', 'VerticalAlignment', 'middle');
end
% Graficar las fuerzas en los nodos seleccionados
for i = 1:length(nodos_fuerza)
nodo = nodos_fuerza(i);
x = nodos(nodo, 1);
y = nodos(nodo, 2);
% Obtener las fuerzas en X y Y
fuerzaX = fuerza_X(i);
```

```
fuerzaY = fuerza_Y(i);
% Graficar una flecha pequeña en el eje X
quiver(x, y, 0.5*(fuerzaX/fuerzaX), 0, 'Color', 'green', 'LineWidth', 1,
'MaxHeadSize', 1);
% Graficar una flecha pequeña en el eje Y
if fuerzaY >= 0
quiver(x, y, 0, 0.5*(fuerzaY/fuerzaY), 'Color', 'cyan', 'LineWidth', 1,
'MaxHeadSize', 1);
else
quiver(x, y, 0, 0.5*(fuerzaY/fuerzaY), 'Color', 'cyan', 'LineWidth', 1,
'MaxHeadSize', 1, 'AutoScale', 'on', 'AutoScaleFactor', -1);
end
% Mostrar los valores de fuerza en X y Y arriba o abajo del nodo
if nodo <= n+1
text(x, y - offset, sprintf('Fx=%.2f\nFy=%.2f', fuerzaX, fuerzaY), 'Color',
'green', 'HorizontalAlignment', 'center', 'VerticalAlignment', 'top');
else
text(x, y + offset, sprintf('Fx=%.2f\nFy=%.2f', fuerzaX, fuerzaY), 'Color',
'green', 'HorizontalAlignment', 'center', 'VerticalAlignment', 'bottom');
end
end
% Crear archivo de texto
fileID = fopen('Datos_Fuerzas.txt', 'w');
% Escribir los datos de las fuerzas en X y Y en el archivo
for i = 1:length(nodos_fuerza)
nodo = nodos_fuerza(i);
fuerzaX = fuerza_X(i);
fuerzaY = fuerza_Y(i);
fprintf(fileID, '%d\t%.2f\t%.2f\n', nodo, fuerzaX, fuerzaY);
end
% Cerrar el archivo
fclose(fileID);
disp('Archivo "Datos_Fuerzas.txt" generado exitosamente.');
% Crear matriz de restricciones
restricciones = zeros(num_nodos, 2);
% Pedir al usuario la cantidad de nodos a los que se les aplicarán
restricciones
num_nodos_restriccion = input('Ingrese la cantidad de nodos a los que desea
aplicar restricciones: ');
% Pedir al usuario los nodos a los que se les aplicarán restricciones
nodos_restriccion = zeros(1, num_nodos_restriccion);
for i = 1:num_nodos_restriccion
nodos_restriccion(i) = input(sprintf('Ingrese el nodo %d al que desea aplicar
restricciones: ', i));
end
% Pedir al usuario las restricciones en X e Y para los nodos seleccionados
for i = 1:num_nodos_restriccion
nodo_restriccion = nodos_restriccion(i);
restriccion_X = input(sprintf('Ingrese la restricción en X para el nodo %d
(0 = desplazamiento, 1 = fijo): ', nodo_restriccion));
restriccion_Y = input(sprintf('Ingrese la restricción en Y para el nodo %d
(0 = desplazamiento, 1 = fijo): ', nodo_restriccion));
restricciones(nodo_restriccion, 1) = restriccion_X;
restricciones(nodo_restriccion, 2) = restriccion_Y;
end
% Aplicar restricciones en X
fuerza_X(restricciones(:, 1) == 1) = NaN;
% Aplicar restricciones en Y
```

```
fuerza_Y(restricciones(:, 2) == 1) = NaN;
% Generar matriz de datos de restricciones
datos_restricciones = [nodos_restriccion' restricciones(nodos_restriccion,
1) restricciones(nodos_restriccion, 2)];
% Generar archivo de texto con las restricciones
fid = fopen('Datos_Restricciones.txt', 'w');
for i = 1:num_nodos_restriccion
fprintf(fid, '%d\t%d\t%d\n', datos_restricciones(i, 1),
datos_restricciones(i, 2), datos_restricciones(i, 3));
end
fclose(fid);
% Agregar etiquetas
xlabel('X');
ylabel('Y');
title('DISEÑO DE TIJERAL');
% Ajustar los ejes para mejor visualización
axis equal;
grid on;
hold off;
% Crear bases de datos para el segundo codigo llamado Truss_TIJERAL.m
% Crear archivo de texto
fileID = fopen('Datos_Nodos.txt', 'w');
% Escribir las coordenadas de cada nodo en el archivo
for i = 1:size(nodos, 1)
fprintf(fileID, '%d %.3f %.3f\n', i, nodos(i, 1), nodos(i, 2));
end
% Cerrar el archivo
fclose(fileID);
disp('Archivo "Datos_Nodos.txt" generado exitosamente.');

% Coordenadas de las barras [nodo1 nodo2]
barras = [];
% Unir el nodo actual con el siguiente nodo
for i = 1:n+1
if i < n+1
nodo_actual = i;
nodo_siguiente = i+1;
barras = [barras; nodo_actual, nodo_siguiente];
end
end
% Unir los nodos 1 con 10, 2 con 12, 3 con 14, 4 con 16, 5 con 18, etc.
for i = 1:n_divisiones+1
idx1 = i;
idx2 = 2 * i + n;
barras = [barras; idx1, idx2];
end
% Unir los nodos 6 con 17, 7 con 15, 8 con 13, etc.
for i = 1:n_divisiones
idx1 = i + n_divisiones + 1;
idx2 = 2*n+3 - (2 * i);
barras = [barras; idx1, idx2];
end
% Unir los nodos 2 con 10, 3 con 12, 4 con 16, 5 con 16, etc.
for i = 1:n_divisiones
idx1 = i + 1;
idx2 = n + (2 * i);
barras = [barras; idx1, idx2];
end
```

```
% Unir los nodos 5 con 17, 6 con 15, 7 con 13, 8 con 11, etc.
for i = 1:n_divisiones
idx1 = i + n_divisiones;
idx2 = 2*n+3 - (2 * i);
barras = [barras; idx1, idx2];
end
% Unir los nodos de la parte izquierda inclinada del tijeral
for i = n+2:2:2*n
nodo_izquierdo = i;
nodo_derecho = i+2;
barras = [barras; nodo_izquierdo, nodo_derecho];
end
% Unir el nodo 2n+2 con el nodo 2n+1
barras = [barras; 2*n+2, 2*n+1];
% Unir los nodos de la parte derecha inclinada del tijeral
for i = 2*n+1:-2:n+5
nodo_izquierdo = i;
nodo_derecho = i-2;
barras = [barras; nodo_izquierdo, nodo_derecho];
end
% Crear archivo de texto Data-barras donde se puede ver las barras y sus
% ubicaciones entre nodos
fileID = fopen('Informacion_Barras.txt', 'w');
% Escribir las coordenadas de las barras y los nodos en el archivo
fprintf(fileID, 'Coordenadas de las barras y los nodos:\n');
for i = 1:size(barras, 1)
nodo1 = barras(i, 1);
nodo2 = barras(i, 2);
x1 = nodos(nodo1, 1);
y1 = nodos(nodo1, 2);
x2 = nodos(nodo2, 1);
y2 = nodos(nodo2, 2);
fprintf(fileID, 'Barra %d: nodo%d (%.3f, %.3f) -> nodo%d (%.3f, %.3f)\n', i,
nodo1, x1, y1, nodo2, x2, y2);
end
% Cerrar el archivo
fclose(fileID);
disp('Archivo "Informacion_Barras.txt" generado exitosamente.');
% Crear el archivo de texto Datos_barras
% Pedir al usuario el área de la barra
area_barra = 0.00000000001;
% Obtener la cantidad de barras
num_barras = size(barras, 1);
% Generar matriz de datos de las barras
datos_barras    =    [transpose(1:num_barras),    barras,    repmat(area_barra,
num_barras, 1), repmat(210000000, num_barras, 1)];
% Generar archivo de texto con los datos de las barras
fid = fopen('Datos_Barras.txt', 'w');
for i = 1:num_barras
fprintf(fid, '%d\t%d\t%d\t%.11f\t%d\n', datos_barras(i, :));
end
fclose(fid);
disp('Archivo "Datos_Barras.txt" generado exitosamente.');
% Calcular el largo de cada barra y almacenarlo en una matriz
longitud_barras = zeros(size(barras, 1), 1);
for i = 1:size(barras, 1)
nodo1 = barras(i, 1);
nodo2 = barras(i, 2);
```

```
x1 = nodos(nodo1, 1);
y1 = nodos(nodo1, 2);
x2 = nodos(nodo2, 1);
y2 = nodos(nodo2, 2);
longitud_barras(i) = sqrt((x2 - x1)^2 + (y2 - y1)^2);
end
% Crear archivo de texto "valoresBarras.txt" solo con las longitudes de cada
barra
fileID = fopen('valoresBarras.txt', 'w');
for i = 1:length(longitud_barras)
fprintf(fileID, '%d\t%.4f\n', i, longitud_barras(i));
end
fclose(fileID);
disp('Archivo "valoresBarras.txt" generado exitosamente.');
% Crear archivo de texto Datos info
num_barras = size(barras, 1);
num_nodos = size(nodos, 1);
num_nodos_fuerzas = length(nodos_fuerza);
num_nodos_restricciones = sum(restricciones(:, 1) == 1 | restricciones(:, 2)
== 1);
data = [num_barras, num_nodos, num_nodos_fuerzas, num_nodos_restricciones];
fid = fopen('Datos_Info.txt', 'w');
fprintf(fid, '%d\t%d\t%d\t%d\n', data);
fclose(fid); disp('Archivo "Datos_Info.txt" generado exitosamente.');
```

Apéndice B Código de programación – Truss_TIJERAL

```
clc
clear all
NFile = input('Ingresar nombre del archivo de Datos = ','s');
NInfo = strcat(NFile, '_Info.txt');
NBarras = strcat(NFile, '_Barras.txt');
NNodos = strcat(NFile, '_Nodos.txt');
NFuerzas = strcat(NFile, '_Fuerzas.txt');
NRestricciones = strcat(NFile, '_Restricciones.txt');
Info = importdata(NInfo);
Barras = importdata(NBarras);
Nodos = importdata(NNodos);
Fuerzas = importdata(NFuerzas);
Restricciones = importdata(NRestricciones);
% Validar que el número de nodos con restricciones no exceda el número total
de nodos
if Info(4) > Info(2)
error('El número de nodos con restricciones excede el número total de
nodos.');
end
% Crear la matriz global
num_nodos = Info(2);
num_barras = Info(1);
K = zeros(2 * num_nodos);
F = zeros(2 * num_nodos, 1);
UG = zeros(2 * num_nodos, 1);
UL = zeros(2 * num_nodos, 1);
L = zeros(num_barras, 1);
C = zeros(num_barras, 1);
S = zeros(num_barras, 1);
CTE = zeros(num_barras, 1);
% Llenar la matriz global
for I = 1:num_barras
L(I) = norm(Nodos(Barras(I, 2), 2:3) - Nodos(Barras(I, 3), 2:3));
C(I) = (Nodos(Barras(I, 2), 2) - Nodos(Barras(I, 3), 2)) / L(I);
S(I) = (Nodos(Barras(I, 2), 3) - Nodos(Barras(I, 3), 3)) / L(I);
CTE(I) = Barras(I, 4) * Barras(I, 5) / L(I);
C2 = CTE(I) * C(I) * C(I);
S2 = CTE(I) * S(I) * S(I);
CS = CTE(I) * C(I) * S(I);
K(2 * Barras(I, 2) - 1, 2 * Barras(I, 2) - 1) = K(2 * Barras(I, 2) - 1, 2 *
Barras(I, 2) - 1) + C2;
K(2 * Barras(I, 2) - 1, 2 * Barras(I, 2)) = K(2 * Barras(I, 2) - 1, 2 *
Barras(I, 2)) + CS;
K(2 * Barras(I, 2), 2 * Barras(I, 2) - 1) = K(2 * Barras(I, 2), 2 * Barras(I,
2) - 1) + CS;
K(2 * Barras(I, 2), 2 * Barras(I, 2)) = K(2 * Barras(I, 2), 2 * Barras(I,
2)) + S2;
K(2 * Barras(I, 3) - 1, 2 * Barras(I, 3) - 1) = K(2 * Barras(I, 3) - 1, 2 *
Barras(I, 3) - 1) + C2;
K(2 * Barras(I, 3) - 1, 2 * Barras(I, 3)) = K(2 * Barras(I, 3) - 1, 2 *
Barras(I, 3)) + CS;
K(2 * Barras(I, 3), 2 * Barras(I, 3) - 1) = K(2 * Barras(I, 3), 2 * Barras(I,
3) - 1) + CS;
K(2 * Barras(I, 3), 2 * Barras(I, 3)) = K(2 * Barras(I, 3), 2 * Barras(I,
3)) + S2;
```

```
K(2 * Barras(I, 2) - 1, 2 * Barras(I, 3) - 1) = K(2 * Barras(I, 2) - 1, 2 *
Barras(I, 3) - 1) - C2;
K(2 * Barras(I, 2) - 1, 2 * Barras(I, 3)) = K(2 * Barras(I, 2) - 1, 2 *
Barras(I, 3)) - CS;
K(2 * Barras(I, 2), 2 * Barras(I, 3) - 1) = K(2 * Barras(I, 2), 2 * Barras(I,
3) - 1) - CS;
K(2 * Barras(I, 2), 2 * Barras(I, 3)) = K(2 * Barras(I, 2), 2 * Barras(I,
3)) - S2;
K(2 * Barras(I, 3) - 1, 2 * Barras(I, 2) - 1) = K(2 * Barras(I, 3) - 1, 2 *
Barras(I, 2) - 1) - C2;
K(2 * Barras(I, 3) - 1, 2 * Barras(I, 2)) = K(2 * Barras(I, 3) - 1, 2 *
Barras(I, 2)) - CS;
K(2 * Barras(I, 3), 2 * Barras(I, 2) - 1) = K(2 * Barras(I, 3), 2 * Barras(I,
2) - 1) - CS;
K(2 * Barras(I, 3), 2 * Barras(I, 2)) = K(2 * Barras(I, 3), 2 * Barras(I,
2)) - S2;
end
% Asignación de fuerzas
for I = 1:Info(3)
F(2 * Fuerzas(I, 1) - 1) = Fuerzas(I, 2);
F(2 * Fuerzas(I, 1)) = Fuerzas(I, 3);
end
% Asignación de restricciones para reducir la matriz K
for I = 1:Info(4)
if Restricciones(I, 2) == 1
UG(2 * Restricciones(I, 1) - 1) = 1;
end
if Restricciones(I, 3) == 1
UG(2 * Restricciones(I, 1)) = 1;
end
end
UG_logical = logical(UG);
K_reduced = K(~UG_logical, ~UG_logical);
F_reduced = F(~UG_logical);
RTA = K_reduced \ F_reduced;
% Volvemos los desplazamientos a su posición
UG(~UG_logical) = RTA;
% Cálculo de fuerzas
FT = zeros(num_barras, 1);
ESF = zeros(num_barras, 1);
for I = 1:num_barras
UL1 = C(I) * UG(2 * Nodos(Barras(I, 2), 1) - 1) + S(I) * UG(2 * Nodos(Barras(I,
2), 1));
UL2 = C(I) * UG(2 * Nodos(Barras(I, 3), 1) - 1) + S(I) * UG(2 * Nodos(Barras(I,
3), 1));
FT(I) = CTE(I) * (UL1 - UL2);
ESF(I) = FT(I) / Barras(I, 4);
end
%Dibujar la estructura
figure(2)
hold on
grid on;
% Calcular escala de las deformaciones
ESCALA = max(L); % Calcular la máxima longitud de la barra
Escaladef = max(abs(UG));
ESCALADEF = ESCALA * 0.05 / Escaladef;
% Poner barras y deformaciones
for I = 1:num_barras
```

```
line([Nodos(Barras(I, 2), 2), Nodos(Barras(I, 3), 2)], [Nodos(Barras(I, 2),
3), Nodos(Barras(I, 3), 3)], 'Color', 'k');
line([Nodos(Barras(I, 2), 2) + UG(2 * Barras(I, 2) - 1) * ESCALADEF,
Nodos(Barras(I, 3), 2) + UG(2 * Barras(I, 3) - 1) * ESCALADEF],
[Nodos(Barras(I, 2), 3) + UG(2 * Barras(I, 2)) * ESCALADEF, Nodos(Barras(I,
3), 3) + UG(2 * Barras(I, 3)) * ESCALADEF], 'Color', 'r');
end
% Poner valores de fuerzas
for I = 1:Info(3)
str = ['Fx=', num2str(Fuerzas(I, 2)), ':Fy=', num2str(Fuerzas(I, 3))];
text(Nodos(Fuerzas(I, 1), 2), Nodos(Fuerzas(I, 1), 3), str,
'HorizontalAlignment', 'left', 'Color', 'b');
end
% Calcular la escala de restricciones
ESCALA = ESCALA * 0.05; % Usar 5% de tamaño
x = -ESCALA:ESCALA/10:ESCALA;
y = sqrt(ESCALA^2 - x.^2);
% Poner restricciones
for I = 1:Info(4)
Xtmp = Nodos(Restricciones(I, 1), 2);
Ytmp = Nodos(Restricciones(I, 1), 3);
plot(x + Xtmp, Ytmp + y, 'b', x + Xtmp, Ytmp - y, 'b');
if Restricciones(I, 2) == 1 && Restricciones(I, 3) == 0
text(Xtmp, Ytmp, 'Y', 'Color', 'b');
elseif Restricciones(I, 2) == 0 && Restricciones(I, 3) == 1
text(Xtmp, Ytmp, 'X', 'Color', 'b');
end
end
axis equal;
grid on;
% Escribir respuesta a un archivo
NRta = strcat(NFile, '_Resultados.txt');
fileID = fopen(NRta, 'w');
for I = 1:num_barras
fprintf(fileID, 'Barra %2d : Fuerza = %.2f\n', Barras(I, 1), FT(I));
fprintf(fileID, 'Barra %2d : Esfuerzo = %.2f\n', Barras(I, 1), ESF(I));
end
for I = 1:num_nodos
fprintf(fileID, 'Nodo %2d : Desplamiento en X = %.6f\n', Nodos(I, 1), UG(2 *
I - 1));
fprintf(fileID, 'Nodo %2d : Desplamiento en Y = %.6f\n', Nodos(I, 1), UG(2 *
I));
end
fclose(fileID);
% Agregar fuerza al archivo valoresBarras.txt
FuerzaBarras = FT; % Utilizamos las fuerzas calculadas previamente (FT)
% Leer el archivo valoresBarras.txt y almacenar los datos en una matriz
valores_barras = dlmread('valoresBarras.txt');
% Agregar la fuerza a la tercera columna de la matriz
valores_barras(:, 3) = FuerzaBarras;
% Crear un nuevo archivo valoresBarras con la fuerza agregada
fileID = fopen('valoresBarras.txt', 'w');
for i = 1:size(valores_barras, 1)
fprintf(fileID, '%d\t%.4f\t%.2f\n', valores_barras(i, 1), valores_barras(i,
2), valores_barras(i, 3));
end
fclose(fileID);
disp('Archivo "valoresBarras.txt" actualizado con la fuerza de cada barra.');
```

Apéndice C Código de programación – Selección Perfil

```
clear all
clc
% Leer el archivo valoresBarras.txt y obtener los valores de fuerza y longitud
data = dlmread('valoresBarras.txt');
% Acceder a los valores de la segunda y tercera columna (columnas de longitud
y fuerza)
longitudes_barras = data(:, 2);
fuerza_barras = data(:, 3);
% Calcular el valor de A_T para cada barra por Tracción
A_T = abs(fuerza_barras) / (0.9 * 2530); % área en cm^2
% Importar datos desde el archivo CSV (AngulosDobles.csv) con el delimitador
';'
filename = 'AngulosDobles.csv';
data = readtable(filename, 'Delimiter', ';', 'ReadVariableNames', false);
% Asignar nombres de columnas a los datos importados
data.Properties.VariableNames = {'Designacion', 'Area', 'rx', 'ry'};
% Convertir la tabla a una matriz numérica y un cell array para el texto
numeric_values = table2array(data(:, 2:end));
text_values = table2cell(data(:, 1));
% Obtener el valor de n_divisiones ingresado por el usuario
n_divisiones = input('Ingresa el valor de n_divisiones = ');
n = 2 * n_divisiones;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Definir los índices para las barras superiores de 3n+2 a 4n+1
indices_barras_superiores = (3 * n + 2):(4 * n + 1);
% Encontrar el índice de la barra con la longitud mayor dentro del rango de
barras superiores
[longitud_mayor, indice_longitud_mayor] =
max(longitudes_barras(indices_barras_superiores));
% Continuamos con el código original para encontrar los valores de r_x y r_y
correspondientes a esta área máxima
% Inicializar vectores para almacenar los valores de r_x y r_y
correspondientes a cada área
r_x_values_superiores = zeros(size(A_T));
r_y_values_superiores = zeros(size(A_T));
% Inicializar variables para almacenar los resultados de las barras superiores
max_A_T_barras_superiores = 0;
max_A_T_indices_superiores = [];
r_x_barras_correspondientes_superiores = [];
r_y_barras_correspondientes_superiores = [];
% Calcular el valor de A_T solo para las barras en el rango de
indices_barras_superiores
A_T_barras_superiores = A_T(indices_barras_superiores);
% Encontrar el valor máximo de A_T en el rango de barras superiores
max_A_T_barras_superiores = max(A_T_barras_superiores);
% Encontrar el índice correspondiente al valor máximo de A_T en el rango de
barras superiores
idx_max_A_T_barras_superiores = find(A_T_barras_superiores ==
max_A_T_barras_superiores, 1);
% Obtener el valor de r_x y r_y correspondiente al máximo A_T en el rango de
barras superiores
r_x_barras_correspondientes_superiores =
numeric_values(idx_max_A_T_barras_superiores, 1); % Primera columna es r_x
r_y_barras_correspondientes_superiores =
numeric_values(idx_max_A_T_barras_superiores, 2); % Segunda columna es r_y
```

```
% Obtener el valor de Lambda (Longitud mayor/r_x correspondiente) para analizar por compresión
Lambda_superiores = floor((longitud_mayor / r_x_barras_correspondientes_superiores) * 100); % se multiplica 100 ya que longitud está en metros y r_x en cm
% Obtener el índice del valor de r_x seleccionado que cumple la condición
idx_r_x_seleccionado_superiores = find(numeric_values(:, 2) == r_x_barras_correspondientes_superiores, 1);
% Obtener los valores de fuerza correspondientes a las barras superiores
fuerza_barras_superiores = fuerza_barras(indices_barras_superiores);
% Calcular la fuerza máxima aplicada en las barras superiores (en valor absoluto)
fuerza_maxima_aplicada_superiores = max(abs(fuerza_barras_superiores));
% Mostrar la fuerza máxima aplicada en las barras superiores
disp(['Fuerza máxima aplicada en las barras superiores: ', num2str(fuerza_maxima_aplicada_superiores), ' kg']);
% Mostrar la designación correspondiente a los valores de r_x y r_y seleccionados por tracción para cada caso
if isempty(idx_r_x_seleccionado_superiores)
disp('No se encontró un r_x válido que cumpla la condición por tracción para las barras superiores.');
% Variables para almacenar los valores de r_x y r_y que cumplan con la condición por compresión
r_x_barras_correspondientes_superiores = 0;
r_y_barras_correspondientes_superiores = 0;
% Realizar el análisis por compresión buscando el valor de r_x que cumpla con la condición
for i = 1:length(numeric_values)
r_x_candidate = numeric_values(i, 2);
Lambda_superiores = floor((longitud_mayor / r_x_candidate) * 100);
if Lambda_superiores <= 200
% Buscar el segundo valor de la segunda columna en el archivo TablasDeEsfCompresion.txt
% (Esto debe hacerse solo una vez)
% Leer el archivo TablasDeEsfCompresion.txt y obtener los datos
filename2 = 'TablasDeEsfCompresion.txt';
data2 = dlmread(filename2);
% Buscar el índice del valor más cercano a Lambda en la primera columna del archivo
[~, idx_lambda] = min(abs(data2(:, 1) - Lambda_superiores));
% Obtener el segundo valor de la segunda columna correspondiente al índice encontrado
segundo_valor = data2(idx_lambda, 2);
% Calcular el área correspondiente al r_x seleccionado
area_barras_superiores = numeric_values(i, 1); % Segunda columna es el área
% Calcular la fuerza de compresión con el nuevo valor de r_x y el área correspondiente
fuerza_compresion_superior = area_barras_superiores * segundo_valor * 1000;
if fuerza_compresion_superior >= fuerza_maxima_aplicada_superiores
r_x_barras_correspondientes_superiores = r_x_candidate;
r_y_barras_correspondientes_superiores = numeric_values(i, 3);
disp('Se ha encontrado un r_x que cumple la condición por compresión para las barras superiores.');
disp(['Nuevo r_x a seleccionar para las barras superiores: ' num2str(r_x_barras_correspondientes_superiores) ' cm']);
disp(['Nuevo r_y a seleccionar para las barras superiores: ' num2str(r_y_barras_correspondientes_superiores) ' cm']);
```

```
disp(['Fuerza de compresión calculada con nuevo r_x para las barras superiores
= ', num2str(fuerza_compresion_superior), ' kg']);
break;
end
end
end
% Mostrar la designación correspondiente a los valores de r_x y r_y
seleccionados por compresión para las barras superiores
if r_x_barras_correspondientes_superiores == 0
disp('No se encontró una designación correspondiente a los valores de r_x y
r_y seleccionados por compresión para las barras superiores.');
else
% Obtener la fila correspondiente a los valores de r_x y r_y seleccionados
idx_fila_seleccionada_superiores = find(numeric_values(:, 2) ==
r_x_barras_correspondientes_superiores & numeric_values(:, 3) ==
r_y_barras_correspondientes_superiores);
% Obtener la designación correspondiente a los valores de r_x y r_y para las
barras superiores
designacion_superiores = text_values{idx_fila_seleccionada_superiores};
disp(['Se usará el tipo de ángulo correspondiente a la designación para las
barras superiores: ', designacion_superiores]);
end
else
% Obtener la designación correspondiente a los valores de r_x y r_y por
tracción para las barras superiores
designacion_superiores = text_values{idx_r_x_seleccionado_superiores};
disp(['Se usará el tipo de ángulo correspondiente a la designación para las
barras superiores: ', designacion_superiores]);
end
disp('--------------------------------------------------------------------
');
% Replicamos el código para las barras inferiores
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% BARRAS INFERIORES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Definir los índices para las barras inferiores de 1 a n
indices_barras_inferiores = 1:n;
% Encontrar el índice de la barra con la longitud mayor dentro del rango de
barras inferiores
[longitud_mayor_inferiores, indice_longitud_mayor_inferiores] =
max(longitudes_barras(indices_barras_inferiores));
% Continuamos con el código original para encontrar los valores de r_x y r_y
correspondientes a esta área máxima
% Inicializar vectores para almacenar los valores de r_x y r_y
correspondientes a cada área
r_x_values_inferiores = zeros(size(A_T));
r_y_values_inferiores = zeros(size(A_T));
% Inicializar variables para almacenar los resultados de las barras inferiores
max_A_T_barras_inferiores = 0;
max_A_T_indices_inferiores = [];
r_x_barras_correspondientes_inferiores = [];
r_y_barras_correspondientes_inferiores = [];
% Calcular el valor de A_T solo para las barras en el rango de
indices_barras_inferiores
A_T_barras_inferiores = A_T(indices_barras_inferiores);
% Encontrar el valor máximo de A_T en el rango de barras inferiores
max_A_T_barras_inferiores = max(A_T_barras_inferiores);
% Encontrar el índice correspondiente al valor máximo de A_T en el rango de
barras inferiores
```

```
idx_max_A_T_barras_inferiores = find(A_T_barras_inferiores == max_A_T_barras_inferiores, 1);
% Obtener el valor de r_x y r_y correspondiente al máximo A_T en el rango de barras inferiores
r_x_barras_correspondientes_inferiores = numeric_values(idx_max_A_T_barras_inferiores, 1); % Primera columna es r_x
r_y_barras_correspondientes_inferiores = numeric_values(idx_max_A_T_barras_inferiores, 2); % Segunda columna es r_y
% Obtener el valor de Lambda (Longitud mayor/r_x correspondiente) para analizar por compresión
Lambda_inferiores = floor((longitud_mayor_inferiores / r_x_barras_correspondientes_inferiores) * 100); % se multiplica 100 ya que longitud está en metros y r_x en cm
% Obtener el índice del valor de r_x seleccionado que cumple la condición
idx_r_x_seleccionado_inferiores = find(numeric_values(:, 2) == r_x_barras_correspondientes_inferiores, 1);
% Obtener los valores de fuerza correspondientes a las barras inferiores
fuerza_barras_inferiores = fuerza_barras(indices_barras_inferiores);
% Calcular la fuerza máxima aplicada en las barras inferiores (en valor absoluto)
fuerza_maxima_aplicada_inferiores = max(abs(fuerza_barras_inferiores));
% Mostrar la fuerza máxima aplicada en las barras inferiores
disp(['Fuerza máxima aplicada en las barras inferiores: ', num2str(fuerza_maxima_aplicada_inferiores), ' kg']);
if isempty(idx_r_x_seleccionado_inferiores)
disp('No se encontró un r_x válido que cumpla la condición por tracción para las barras inferiores.');
% Variables para almacenar los valores de r_x y r_y que cumplan con la condición por compresión
r_x_barras_correspondientes_inferiores = 0;
r_y_barras_correspondientes_inferiores = 0;
% Realizar el análisis por compresión buscando el valor de r_x que cumpla con la condición
for i = 1:length(numeric_values)
r_x_candidate = numeric_values(i, 2);
Lambda_inferiores = floor((longitud_mayor_inferiores / r_x_candidate) * 100);
if Lambda_inferiores <= 200
% Buscar el segundo valor de la segunda columna en el archivo TablasDeEsfCompresion.txt
% (Esto debe hacerse solo una vez)
% Leer el archivo TablasDeEsfCompresion.txt y obtener los datos
filename2 = 'TablasDeEsfCompresion.txt';
data2 = dlmread(filename2);
% Buscar el índice del valor más cercano a Lambda en la primera columna del archivo
[~, idx_lambda] = min(abs(data2(:, 1) - Lambda_inferiores));
% Obtener el segundo valor de la segunda columna correspondiente al índice encontrado
segundo_valor = data2(idx_lambda, 2);
% Calcular el área correspondiente al r_x seleccionado
area_barras_inferiores = numeric_values(i, 1); % Segunda columna es el área
% Calcular la fuerza de compresión con el nuevo valor de r_x y el área correspondiente
fuerza_compresion_inferior = area_barras_inferiores * segundo_valor * 1000;
if fuerza_compresion_inferior >= fuerza_maxima_aplicada_inferiores
r_x_barras_correspondientes_inferiores = r_x_candidate;
r_y_barras_correspondientes_inferiores = numeric_values(i, 3);
```

```
disp('Se ha encontrado un r_x que cumple la condición por compresión para las barras inferiores.');
disp(['Nuevo r_x a seleccionar para las barras inferiores: ' num2str(r_x_barras_correspondientes_inferiores) ' cm']);
disp(['Nuevo r_y a seleccionar para las barras inferiores: ' num2str(r_y_barras_correspondientes_inferiores) ' cm']);
disp(['Fuerza de compresión calculada con nuevo r_x para las barras inferiores = ', num2str(fuerza_compresion_inferior), ' kg']);
break;
end
end
end
% Mostrar la designación correspondiente a los valores de r_x y r_y seleccionados por compresión para las barras inferiores
if r_x_barras_correspondientes_inferiores == 0
disp('No se encontró una designación correspondiente a los valores de r_x y r_y seleccionados por compresión para las barras inferiores.');
else
% Obtener la fila correspondiente a los valores de r_x y r_y seleccionados
idx_fila_seleccionada_inferiores = find(numeric_values(:, 2) == r_x_barras_correspondientes_inferiores & numeric_values(:, 3) == r_y_barras_correspondientes_inferiores);
% Obtener la designación correspondiente a los valores de r_x y r_y para las barras inferiores
designacion_inferiores = text_values{idx_fila_seleccionada_inferiores};
disp(['Se usará el tipo de ángulo correspondiente a la designación para las barras inferiores: ', designacion_inferiores]);
end
else
% Obtener la designación correspondiente a los valores de r_x y r_y por tracción para las barras inferiores
designacion_inferiores = text_values{idx_r_x_seleccionado_inferiores};
disp(['Se usará el tipo de ángulo correspondiente a la designación para las barras inferiores: ', designacion_inferiores]);
end
disp('--------------------------------------------------------------------
');
% Replicamos el código para las barras diagonales
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% BARRAS DIAGONALES %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Definir los índices para las barras diagonales de 2n+2 a 3n+1
indices_barras_diagonales = (2 * n + 2):(3 * n + 1);
% Encontrar el índice de la barra con la longitud mayor dentro del rango de barras diagonales
[longitud_mayor_diagonales, indice_longitud_mayor_diagonales] = max(longitudes_barras(indices_barras_diagonales));
% Continuamos con el código original para encontrar los valores de r_x y r_y correspondientes a esta área máxima
% Inicializar vectores para almacenar los valores de r_x y r_y correspondientes a cada área
r_x_values_diagonales = zeros(size(A_T));
r_y_values_diagonales = zeros(size(A_T));
% Inicializar variables para almacenar los resultados de las barras diagonales
max_A_T_barras_diagonales = 0;
max_A_T_indices_diagonales = [];
r_x_barras_correspondientes_diagonales = [];
r_y_barras_correspondientes_diagonales = [];
```

```
% Calcular el valor de A_T solo para las barras en el rango de indices_barras_diagonales
A_T_barras_diagonales = A_T(indices_barras_diagonales);
% Encontrar el valor máximo de A_T en el rango de barras diagonales
max_A_T_barras_diagonales = max(A_T_barras_diagonales);
% Encontrar el índice correspondiente al valor máximo de A_T en el rango de barras diagonales
idx_max_A_T_barras_diagonales = find(A_T_barras_diagonales == max_A_T_barras_diagonales, 1);
% Obtener el valor de r_x y r_y correspondiente al máximo A_T en el rango de barras diagonales
r_x_barras_correspondientes_diagonales = numeric_values(idx_max_A_T_barras_diagonales, 1); % Primera columna es r_x
r_y_barras_correspondientes_diagonales = numeric_values(idx_max_A_T_barras_diagonales, 2); % Segunda columna es r_y
% Obtener el valor de Lambda (Longitud mayor/r_x correspondiente) para analizar por compresión
Lambda_diagonales = floor((longitud_mayor_diagonales / r_x_barras_correspondientes_diagonales) * 100); % se multiplica 100 ya que longitud está en metros y r_x en cm
% Obtener el índice del valor de r_x seleccionado que cumple la condición
idx_r_x_seleccionado_diagonales = find(numeric_values(:, 2) == r_x_barras_correspondientes_diagonales, 1);
% Obtener los valores de fuerza correspondientes a las barras diagonales
fuerza_barras_diagonales = fuerza_barras(indices_barras_diagonales);
% Calcular la fuerza máxima aplicada en las barras diagonales (en valor absoluto)
fuerza_maxima_aplicada_diagonales = max(abs(fuerza_barras_diagonales));
% Mostrar la fuerza máxima aplicada en las barras diagonales
disp(['Fuerza máxima aplicada en las barras diagonales: ', num2str(fuerza_maxima_aplicada_diagonales), ' kg']);
if isempty(idx_r_x_seleccionado_diagonales)
disp('No se encontró un r_x válido que cumpla la condición por tracción para las barras diagonales.');
% Variables para almacenar los valores de r_x y r_y que cumplan con la condición por compresión
r_x_barras_correspondientes_diagonales = 0;
r_y_barras_correspondientes_diagonales = 0;
% Realizar el análisis por compresión buscando el valor de r_x que cumpla con la condición
for i = 1:length(numeric_values)
r_x_candidate = numeric_values(i, 2);
Lambda_diagonales = floor((longitud_mayor_diagonales / r_x_candidate) * 100);
if Lambda_diagonales <= 200
% Buscar el segundo valor de la segunda columna en el archivo TablasDeEsfCompresion.txt
% (Esto debe hacerse solo una vez)
% Leer el archivo TablasDeEsfCompresion.txt y obtener los datos
filename2 = 'TablasDeEsfCompresion.txt';
data2 = dlmread(filename2);
% Buscar el índice del valor más cercano a Lambda en la primera columna del archivo
[~, idx_lambda] = min(abs(data2(:, 1) - Lambda_diagonales));
% Obtener el segundo valor de la segunda columna correspondiente al índice encontrado
segundo_valor = data2(idx_lambda, 2);
% Calcular el área correspondiente al r_x seleccionado
area_barras_diagonales = numeric_values(i, 1); % Segunda columna es el área
```

```
% Calcular la fuerza de compresión con el nuevo valor de r_x y el área
correspondiente
fuerza_compresion_diagonal = area_barras_diagonales * segundo_valor * 1000;
if fuerza_compresion_diagonal >= fuerza_maxima_aplicada_diagonales
r_x_barras_correspondientes_diagonales = r_x_candidate;
r_y_barras_correspondientes_diagonales = numeric_values(i, 3);
disp('Se ha encontrado un r_x que cumple la condición por compresión para
las barras diagonales.');
disp(['Nuevo r_x a seleccionar para las barras diagonales: '
num2str(r_x_barras_correspondientes_diagonales) ' cm']);
disp(['Nuevo r_y a seleccionar para las barras diagonales: '
num2str(r_y_barras_correspondientes_diagonales) ' cm']);
disp(['Fuerza de compresión calculada con nuevo r_x para las barras diagonales
= ', num2str(fuerza_compresion_diagonal), ' kg']);
break;
end
end
end
% Mostrar la designación correspondiente a los valores de r_x y r_y
seleccionados por compresión para las barras diagonales
if r_x_barras_correspondientes_diagonales == 0
disp('No se encontró una designación correspondiente a los valores de r_x y
r_y seleccionados por compresión para las barras diagonales.');
else
% Obtener la fila correspondiente a los valores de r_x y r_y seleccionados
idx_fila_seleccionada_diagonales = find(numeric_values(:, 2) ==
r_x_barras_correspondientes_diagonales & numeric_values(:, 3) ==
r_y_barras_correspondientes_diagonales);
% Obtener la designación correspondiente a los valores de r_x y r_y para las
barras diagonales
designacion_diagonales = text_values{idx_fila_seleccionada_diagonales};
disp(['Se usará el tipo de ángulo correspondiente a la designación para las
barras diagonales: ', designacion_diagonales]);
end
else
% Obtener la designación correspondiente a los valores de r_x y r_y por
tracción para las barras diagonales
designacion_diagonales = text_values{idx_r_x_seleccionado_diagonales};
disp(['Se usará el tipo de ángulo correspondiente a la designación para las
barras diagonales: ', designacion_diagonales]);
end
disp('-----------------------------------------------------------------
');
% Replicamos el código para las barras internas perpendiculares
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Definir los índices para las barras internas perpendiculares de n+2 a 2n
indices_barras_int_perpendiculares = (n + 2):(2 * n);
% Encontrar el índice de la barra con la longitud mayor dentro del rango de
barras internas perpendiculares
[longitud_mayor_int_perpendiculares,
indice_longitud_mayor_int_perpendiculares] =
max(longitudes_barras(indices_barras_int_perpendiculares));
% Continuamos con el código original para encontrar los valores de r_x y r_y
correspondientes a esta área máxima
% Inicializar vectores para almacenar los valores de r_x y r_y
correspondientes a cada área
r_x_values_int_perpendiculares = zeros(size(A_T));
```

```
r_y_values_int_perpendiculares = zeros(size(A_T));
% Inicializar variables para almacenar los resultados de las barras internas perpendiculares
max_A_T_barras_int_perpendiculares = 0;
max_A_T_indices_int_perpendiculares = [];
r_x_barras_correspondientes_int_perpendiculares = [];
r_y_barras_correspondientes_int_perpendiculares = [];
% Calcular el valor de A_T solo para las barras en el rango de indices_barras_int_perpendiculares
A_T_barras_int_perpendiculares = A_T(indices_barras_int_perpendiculares);
% Encontrar el valor máximo de A_T en el rango de barras internas perpendiculares
max_A_T_barras_int_perpendiculares = max(A_T_barras_int_perpendiculares);
% Encontrar el índice correspondiente al valor máximo de A_T en el rango de barras internas perpendiculares
idx_max_A_T_barras_int_perpendiculares = find(A_T_barras_int_perpendiculares == max_A_T_barras_int_perpendiculares, 1);
% Obtener el valor de r_x y r_y correspondiente al máximo A_T en el rango de barras internas perpendiculares
r_x_barras_correspondientes_int_perpendiculares = numeric_values(idx_max_A_T_barras_int_perpendiculares, 1); % Primera columna es r_x
r_y_barras_correspondientes_int_perpendiculares = numeric_values(idx_max_A_T_barras_int_perpendiculares, 2); % Segunda columna es r_y
% Obtener el valor de Lambda (Longitud mayor/r_x correspondiente) para analizar por compresión
Lambda_int_perpendiculares = floor((longitud_mayor_int_perpendiculares / r_x_barras_correspondientes_int_perpendiculares) * 100); % se multiplica 100 ya que longitud está en metros y r_x en cm
% Obtener el índice del valor de r_x seleccionado que cumple la condición
idx_r_x_seleccionado_int_perpendiculares = find(numeric_values(:, 2) == r_x_barras_correspondientes_int_perpendiculares, 1);
% Obtener los valores de fuerza correspondientes a las barras internas perpendiculares
fuerza_barras_int_perpendiculares = fuerza_barras(indices_barras_int_perpendiculares);
% Calcular la fuerza máxima aplicada en las barras internas perpendiculares (en valor absoluto)
fuerza_maxima_aplicada_int_perpendiculares = max(abs(fuerza_barras_int_perpendiculares));
% Mostrar la fuerza máxima aplicada en las barras internas perpendiculares
disp(['Fuerza máxima aplicada en las barras internas perpendiculares: ', num2str(fuerza_maxima_aplicada_int_perpendiculares), ' kg']);
if isempty(idx_r_x_seleccionado_int_perpendiculares)
disp('No se encontró un r_x válido que cumpla la condición por tracción para las barras internas perpendiculares.');
% Variables para almacenar los valores de r_x y r_y que cumplan con la condición por compresión
r_x_barras_correspondientes_int_perpendiculares = 0;
r_y_barras_correspondientes_int_perpendiculares = 0;
% Realizar el análisis por compresión buscando el valor de r_x que cumpla con la condición
for i = 1:length(numeric_values)
r_x_candidate = numeric_values(i, 2);
Lambda_int_perpendiculares = floor((longitud_mayor_int_perpendiculares / r_x_candidate) * 100);
if Lambda_int_perpendiculares <= 200
```

```
% Buscar el segundo valor de la segunda columna en el archivo
TablasDeEsfCompresion.txt
% (Esto debe hacerse solo una vez)
% Leer el archivo TablasDeEsfCompresion.txt y obtener los datos
filename2 = 'TablasDeEsfCompresion.txt';
data2 = dlmread(filename2);
% Buscar el índice del valor más cercano a Lambda en la primera columna del
archivo
[~, idx_lambda] = min(abs(data2(:, 1) - Lambda_int_perpendiculares));
% Obtener el segundo valor de la segunda columna correspondiente al índice
encontrado
segundo_valor = data2(idx_lambda, 2);
% Calcular el área correspondiente al r_x seleccionado
area_barras_int_perpendiculares = numeric_values(i, 1); % Segunda columna es
el área
% Calcular la fuerza de compresión con el nuevo valor de r_x y el área
correspondiente
fuerza_compresion_perpendicular = area_barras_int_perpendiculares *
segundo_valor * 1000;
if fuerza_compresion_perpendicular >=
fuerza_maxima_aplicada_int_perpendiculares
r_x_barras_correspondientes_int_perpendiculares = r_x_candidate;
r_y_barras_correspondientes_int_perpendiculares = numeric_values(i, 3);
disp('Se ha encontrado un r_x que cumple la condición por compresión para
las barras internas perpendiculares.');
disp(['Nuevo r_x a seleccionar para las barras internas perpendiculares: '
num2str(r_x_barras_correspondientes_int_perpendiculares) ' cm']);
disp(['Nuevo r_y a seleccionar para las barras internas perpendiculares: '
num2str(r_y_barras_correspondientes_int_perpendiculares) ' cm']);
disp(['Fuerza de compresión calculada con nuevo r_x para las barras internas
perpendiculares = ', num2str(fuerza_compresion_perpendicular), ' kg']);
break;
end
end
end
% Mostrar la designación correspondiente a los valores de r_x y r_y
seleccionados por compresión para las barras internas perpendiculares
if r_x_barras_correspondientes_int_perpendiculares == 0
disp('No se encontró una designación correspondiente a los valores de r_x y
r_y seleccionados por compresión para las barras internas perpendiculares.');
else
% Obtener la fila correspondiente a los valores de r_x y r_y seleccionados
idx_fila_seleccionada_int_perpendiculares = find(numeric_values(:, 2) ==
r_x_barras_correspondientes_int_perpendiculares & numeric_values(:, 3) ==
r_y_barras_correspondientes_int_perpendiculares);
% Obtener la designación correspondiente a los valores de r_x y r_y para las
barras internas perpendiculares
designacion_int_perpendiculares =
text_values{idx_fila_seleccionada_int_perpendiculares};
disp(['Se usará el tipo de ángulo correspondiente a la designación para las
barras internas perpendiculares: ', designacion_int_perpendiculares]);
end
else
% Obtener la designación correspondiente a los valores de r_x y r_y por
tracción para las barras internas perpendiculares
designacion_int_perpendiculares =
text_values{idx_r_x_seleccionado_int_perpendiculares};
```

```
disp(['Se usará el tipo de ángulo correspondiente a la designación para las barras internas perpendiculares: ', designacion_int_perpendiculares]);
end
disp('----------------------------------------------------------------------
');
% Replicamos el código para las barras laterales
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% BARRAS LATERALES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Definir los índices para las barras laterales, que en este caso es un solo índice n+1
indice_barras_laterales = n + 1;
% Encontrar el valor de r_x y r_y correspondiente a estas barras laterales
r_x_barras_correspondientes_laterales = numeric_values(indice_barras_laterales, 1); % Primera columna es r_x
r_y_barras_correspondientes_laterales = numeric_values(indice_barras_laterales, 2); % Segunda columna es r_y
% Obtener el valor de Lambda (Longitud mayor/r_x correspondiente) para analizar por compresión
Lambda_laterales = floor((longitudes_barras(indice_barras_laterales) / r_x_barras_correspondientes_laterales) * 100); % se multiplica 100 ya que longitud está en metros y r_x en cm
% Obtener el índice del valor de r_x seleccionado que cumple la condición
idx_r_x_seleccionado_laterales = find(numeric_values(:, 2) == r_x_barras_correspondientes_laterales, 1);
% Obtener los valores de fuerza correspondientes a las barras laterales
fuerza_barras_laterales = fuerza_barras(indice_barras_laterales);
% Calcular la fuerza máxima aplicada en las barras laterales (en valor absoluto)
fuerza_maxima_aplicada_laterales = max(abs(fuerza_barras_laterales));
% Mostrar la fuerza máxima aplicada en las barras laterales
disp(['Fuerza máxima aplicada en las barras laterales: ', num2str(fuerza_maxima_aplicada_laterales), ' kg']);
if isempty(idx_r_x_seleccionado_laterales)
disp('No se encontró un r_x válido que cumpla la condición por tracción para las barras laterales.');
% Variables para almacenar los valores de r_x y r_y que cumplan con la condición por compresión
r_x_barras_correspondientes_laterales = 0;
r_y_barras_correspondientes_laterales = 0;
% Realizar el análisis por compresión buscando el valor de r_x que cumpla con la condición
for i = 1:length(numeric_values)
r_x_candidate = numeric_values(i, 2);
Lambda_laterales = floor((longitudes_barras(indice_barras_laterales) / r_x_candidate) * 100);
if Lambda_laterales <= 200
% Buscar el segundo valor de la segunda columna en el archivo TablasDeEsfCompresion.txt
% (Esto debe hacerse solo una vez)
% Leer el archivo TablasDeEsfCompresion.txt y obtener los datos
filename2 = 'TablasDeEsfCompresion.txt';
data2 = dlmread(filename2);
% Buscar el índice del valor más cercano a Lambda en la primera columna del archivo
[~, idx_lambda] = min(abs(data2(:, 1) - Lambda_laterales));
% Obtener el segundo valor de la segunda columna correspondiente al índice encontrado
segundo_valor = data2(idx_lambda, 2);
```

```
% Calcular el área correspondiente al r_x seleccionado
area_barras_laterales = numeric_values(i, 1); % Segunda columna es el área
% Calcular la fuerza de compresión con el nuevo valor de r_x y el área
correspondiente
fuerza_compresion_lateral = area_barras_laterales * segundo_valor * 1000;
if fuerza_compresion_lateral >= fuerza_maxima_aplicada_laterales
r_x_barras_correspondientes_laterales = r_x_candidate;
r_y_barras_correspondientes_laterales = numeric_values(i, 3);
disp('Se ha encontrado un r_x que cumple la condición por compresión para
las barras laterales.');
disp(['Nuevo r_x a seleccionar para las barras laterales: '
num2str(r_x_barras_correspondientes_laterales) ' cm']);
disp(['Nuevo r_y a seleccionar para las barras laterales: '
num2str(r_y_barras_correspondientes_laterales) ' cm']);
disp(['Fuerza de compresión calculada con nuevo r_x para las barras laterales
= ', num2str(fuerza_compresion_lateral), ' kg']);
break;
end
end
end
% Mostrar la designación correspondiente a los valores de r_x y r_y
% seleccionados por compresión para las barra laterales
if r_x_barras_correspondientes_laterales == 0
disp('No se encontró una designación correspondiente a los valores de r_x y
r_y seleccionados por compresión para las barras laterales.');
else
% Obtener la fila correspondiente a los valores de r_x y r_y seleccionados
idx_fila_seleccionada_laterales = find(numeric_values(:, 2) ==
r_x_barras_correspondientes_laterales & numeric_values(:, 3) ==
r_y_barras_correspondientes_laterales);
% Obtener la designación correspondiente a los valores de r_x y r_y para las
barras laterales
designacion_laterales = text_values{idx_fila_seleccionada_laterales};
disp(['Se usará el tipo de ángulo correspondiente a la designación para las
barras laterales: ', designacion_laterales]);
end
else
% Obtener la designación correspondiente a los valores de r_x y r_y por
tracción para las barras laterales
designacion_laterales = text_values{idx_r_x_seleccionado_laterales};
disp(['Se usará el tipo de ángulo correspondiente a la designación para las
barras laterales : ', designacion_laterales]);
end
%Generar archivo con las especificaciones para la selección del ángulo
% Nombre del archivo
filename = 'InformacionResultante.txt';
% Abrir el archivo para escritura
fileID = fopen(filename, 'w');
% Escribir los resultados para las barras superiores
fprintf(fileID, 'RESULTADOS PARA LAS BARRAS SUPERIORES: \n');
fprintf(fileID, 'Fuerza máxima aplicada: %f kg \n',
fuerza_maxima_aplicada_superiores);
fprintf(fileID, 'Fuerza de compresion dado el angulo seleccioando: %f kg\n',
fuerza_compresion_superior);
fprintf(fileID, ['Valores seleccionados para las barras superiores: r_x = ',
num2str(r_x_barras_correspondientes_superiores), ' cm, r_y = ',
num2str(r_y_barras_correspondientes_superiores), ' cm\n']);
```

```
fprintf(fileID, 'Ángulo seleccionado por compresión: %s\n',
designacion_superiores);
fprintf(fileID, '\n');
% Escribir los resultados para las barras inferiores
fprintf(fileID, 'RESULTADOS PARA LAS BARRAS INFERIORES: \n');
fprintf(fileID, 'Fuerza máxima aplicada: %f kg\n',
fuerza_maxima_aplicada_inferiores);
fprintf(fileID, 'Fuerza de compresion dado el angulo seleccionado: %f kg\n',
fuerza_compresion_inferior);
fprintf(fileID, ['Valores seleccionados para las barras inferiores: r_x = ',
num2str(r_x_barras_correspondientes_inferiores), ' cm, r_y = ',
num2str(r_y_barras_correspondientes_inferiores), ' cm\n']);
fprintf(fileID, 'Ángulo seleccionado por compresión: %s\n',
designacion_inferiores);
fprintf(fileID, '\n');
% Escribir los resultados para las barras diagonales
fprintf(fileID, 'RESULTADOS PARA LAS BARRAS DIAGONALES: \n');
fprintf(fileID, 'Fuerza máxima aplicada: %f kg\n',
fuerza_maxima_aplicada_diagonales);
fprintf(fileID, 'Fuerza de compresion dado el angulo seleccionado: %f kg\n',
fuerza_compresion_diagonal);
fprintf(fileID, ['Valores seleccionados para las barras diagonales: r_x = ',
num2str(r_x_barras_correspondientes_diagonales), ' cm, r_y = ',
num2str(r_y_barras_correspondientes_diagonales), ' cm\n']);
fprintf(fileID, 'Ángulo seleccionado por compresión: %s\n',
designacion_diagonales);
fprintf(fileID, '\n');
% Escribir los resultados para las barras perpendiculares internas
fprintf(fileID, 'RESUSLTADOS PARA LAS BARRAS INTERNAS PERPENDICULARES:\n');
fprintf(fileID, 'Fuerza máxima aplicada: %f kg\n',
fuerza_maxima_aplicada_int_perpendiculares);
fprintf(fileID, 'Fuerza de compresion dado el angulo seleccionado: %f kg\n',
fuerza_compresion_perpendicular);
fprintf(fileID,['Valores seleccionados para las barras internas
perpendiculares: r_x = ',
num2str(r_x_barras_correspondientes_int_perpendiculares), ' cm, r_y = ',
num2str(r_y_barras_correspondientes_int_perpendiculares), ' cm\n']);
fprintf(fileID, 'Ángulo seleccionado por compresión: %s\n',
designacion_int_perpendiculares);
fprintf(fileID, '\n');
% Escribir los resultados para las barras laterales
fprintf(fileID, 'RESULTADO PARA LAS BARRAS LATERALES:\n');
fprintf(fileID, 'Fuerza máxima aplicada: %f kg\n',
fuerza_maxima_aplicada_laterales);
fprintf(fileID, 'Fuerza de compresion dado el angulo seleccionado: %f kg\n',
fuerza_compresion_lateral);
fprintf(fileID, ['Valores seleccionados para las barras laterales: r_x = ',
num2str(r_x_barras_correspondientes_laterales), ' cm, r_y = ',
num2str(r_y_barras_correspondientes_laterales), ' cm\n']);
fprintf(fileID, 'Ángulo seleccionado por compresión: %s\n',
designacion_laterales);
fprintf(fileID, '\n');
% Cierra el archivo
fclose(fileID);
% Mostrar los valores de r_x y r_y seleccionados para cada tipo de barra
disp('-----------------------------------------------------------------
');
```

```
disp(['Valores seleccionados para las barras superiores: r_x = ',
num2str(r_x_barras_correspondientes_superiores), ' cm, r_y = ',
num2str(r_y_barras_correspondientes_superiores), ' cm']);
disp(['Valores seleccionados para las barras inferiores: r_x = ',
num2str(r_x_barras_correspondientes_inferiores), ' cm, r_y = ',
num2str(r_y_barras_correspondientes_inferiores), ' cm']);
disp(['Valores seleccionados para las barras diagonales: r_x = ',
num2str(r_x_barras_correspondientes_diagonales), ' cm, r_y = ',
num2str(r_y_barras_correspondientes_diagonales), ' cm']);
disp(['Valores seleccionados para las barras internas perpendiculares: r_x =
', num2str(r_x_barras_correspondientes_int_perpendiculares), ' cm, r_y = ',
num2str(r_y_barras_correspondientes_int_perpendiculares), ' cm']);
disp(['Valores seleccionados para las barras laterales: r_x = ',
num2str(r_x_barras_correspondientes_laterales) ' cm, r_y = ',
num2str(r_y_barras_correspondientes_laterales), ' cm']);
disp('-----------------------------------------------------------------
');
% Mostrar la designación correspondiente para cada tipo de barra
disp('-----------------------------------------------------------------
');
disp(['Designación para las barras superiores: ', designacion_superiores]);
disp(['Designación para las barras inferiores: ', designacion_inferiores]);
disp(['Designación para las barras diagonales: ', designacion_diagonales]);
disp(['Designación para las barras internas perpendiculares: ',
designacion_int_perpendiculares]);
disp(['Designación para las barras laterales: ', designacion_laterales]);
disp('-----------------------------------------------------------------
');
% Calcular la fuerza en Newtons (valor absoluto de la segunda columna
multiplicado por 9.81)
fuerza_newtons = abs(fuerza_barras) * 9.81;
% Leer el archivo valoresBarras.txt y almacenar los datos en una matriz
valores_barras = dlmread('valoresBarras.txt');
% Acceder a los valores de la segunda y tercera columna (columnas de longitud
y fuerza)
longitudes_barras = valores_barras(:, 2);
fuerza_barras = valores_barras(:, 3);
% Agregar las columnas de fuerza en Newtons y fuerza/Area al final de la
matriz
valores_barras(:, 4) = fuerza_newtons;
% Calcular el esfuerzo solo para las barras superiores
indices_barras_superiores = (3 * n + 2):(4 * n + 1);
esfuerzo_barras_superiores = fuerza_newtons(indices_barras_superiores)
./(area_barras_superiores/10000);
% Calcular el esfuerzo solo para las barras inferiores
indices_barras_inferiores = 1:n;
esfuerzo_barras_inferiores = fuerza_newtons(indices_barras_inferiores)
./(area_barras_inferiores/10000);
% Calcular el esfuerzo solo para las barras diagonales
indices_barras_diagonales = (2* n + 2):(3 * n + 1);
esfuerzo_barras_diagonales = fuerza_newtons(indices_barras_diagonales)
./(area_barras_diagonales/10000);
% Calcular el esfuerzo solo para las barras internas perpendiculares
indices_barras_int_perpendiculares = ( n + 2):(2 * n );
esfuerzo_barras_int_perpendiculares =
fuerza_newtons(indices_barras_int_perpendiculares)
./(area_barras_int_perpendiculares/10000);
% Calcular el esfuerzo solo para las barras laterales
```

```
indices_barras_laterales = n+1;
esfuerzo_barras_laterales     =     fuerza_newtons(indices_barras_laterales)
./(area_barras_laterales/10000);
% Agregar las columnas de fuerza en Newtons y esfuerzo al final de la matriz
valores_barras(:, 4) = fuerza_newtons;
valores_barras(:, 5) = 0; % Inicializar el esfuerzo para todas las barras
como cero
valores_barras(indices_barras_superiores, 5) = esfuerzo_barras_superiores;
valores_barras(indices_barras_inferiores, 5) = esfuerzo_barras_inferiores;
valores_barras(indices_barras_diagonales, 5) = esfuerzo_barras_diagonales;
valores_barras(indices_barras_int_perpendiculares,           5)           =
esfuerzo_barras_int_perpendiculares;
valores_barras(indice_barras_laterales, 5) = esfuerzo_barras_laterales;
% Crear un nuevo archivo valoresBarras con la fuerza y esfuerzo agregados
fileID = fopen('valoresBarrasActualizado.txt', 'w');
for i = 1:size(valores_barras, 1)
fprintf(fileID, '%d\t %.3f\t %.2f\t %.2f\t %.2f \n', valores_barras(i, 1),
valores_barras(i,   2),   valores_barras(i,   3),   valores_barras(i,   4),
valores_barras(i, 5));
end
fclose(fileID);
disp('Archivo "valoresBarrasActualizado.txt" actualizado con la fuerza y
esfuerzo de cada barra.');
% Abrir el archivo de texto para lectura
fid = fopen('ValoresBarrasActualizado.txt', 'r');
% Leer los datos del archivo de texto
datos = textscan(fid, '%s %s %s', 'Delimiter', ',');
% Cerrar el archivo de texto
fclose(fid);
% Convertir los datos a una tabla
tablaDatos      =      table(valores_barras(:,1),      valores_barras(:,2),
valores_barras(:,3),      valores_barras(:,4),      valores_barras(:,5),
'VariableNames', {'N° Barra', 'Longitud (m)', 'Fuerza (kg)', 'Fuerza (N)',
'Esfuerzo (Pa)'});
% Escribir los datos a un archivo Excel
writetable(tablaDatos, 'ValoresBarrasActualizado.xlsx');
```

Anexos

Anexo A Tablas de esfuerzos de diseño para miembros en compresión

TABLA DE ESFUERZOS DE DISEÑO PARA MIEMBROS EN COMPRESION

ACERO $F_y = 2530\ kg/cm^2$

$(\phi_c = 0.85)$

Kl/r	$\phi_c F_{cr}$ ksi	$\phi_c F_{cr}$ t/cm²	Kl/r	$\phi_c F_{cr}$ ksi	$\phi_c F_{cr}$ t/cm²	Kl/r	$\phi_c F_{cr}$ ksi	$\phi_c F_{cr}$ t/cm²	Kl/r	$\phi_c F_{cr}$ ksi	$\phi_c F_{cr}$ t/cm²	Kl/r	$\phi_c F_{cr}$ ksi	$\phi_c F_{cr}$ t/cm²
1	30.60	2.16	41	28.01	1.97	81	21.66	1.53	121	14.16	1.00	161	8.23	0.58
2	30.59	2.16	42	27.89	1.96	82	21.48	1.51	122	13.98	0.98	162	8.13	0.57
3	30.59	2.15	43	27.76	1.96	83	21.29	1.50	123	13.80	0.97	163	8.03	0.57
4	30.57	2.15	44	27.63	1.95	84	21.11	1.49	124	13.62	0.96	164	7.93	0.56
5	30.56	2.15	45	27.51	1.94	85	20.92	1.47	125	13.44	0.95	165	7.84	0.55
6	30.54	2.15	46	27.37	1.93	86	20.73	1.46	126	13.27	0.93	166	7.74	0.55
7	30.52	2.15	47	27.24	1.92	87	20.54	1.45	127	13.09	0.92	167	7.65	0.54
8	30.50	2.15	48	27.10	1.91	88	20.35	1.43	128	12.92	0.91	168	7.56	0.53
9	30.47	2.15	49	26.97	1.90	89	20.17	1.42	129	12.74	0.90	169	7.47	0.53
10	30.44	2.14	50	26.83	1.89	90	19.98	1.41	130	12.57	0.89	170	7.38	0.52
11	30.41	2.14	51	26.68	1.88	91	19.79	1.39	131	12.40	0.87	171	7.30	0.51
12	30.37	2.14	52	26.54	1.87	92	19.60	1.38	132	12.23	0.86	172	7.21	0.51
13	30.33	2.14	53	26.39	1.86	93	19.41	1.37	133	12.06	0.85	173	7.13	0.50
14	30.29	2.13	54	26.25	1.85	94	19.22	1.35	134	11.88	0.84	174	7.05	0.50
15	30.24	2.13	55	26.10	1.84	95	19.03	1.34	135	11.71	0.82	175	6.97	0.49
16	30.19	2.13	56	25.94	1.83	96	18.84	1.33	136	11.54	0.81	176	6.89	0.49
17	30.14	2.12	57	25.79	1.82	97	18.65	1.31	137	11.37	0.80	177	6.81	0.48
18	30.08	2.12	58	25.63	1.81	98	18.46	1.30	138	11.20	0.79	178	6.73	0.47
19	30.02	2.12	59	25.48	1.79	99	18.27	1.29	139	11.04	0.78	179	6.66	0.47
20	29.96	2.11	60	25.32	1.78	100	18.08	1.27	140	10.89	0.77	180	6.59	0.46
21	29.90	2.11	61	25.16	1.77	101	17.89	1.26	141	10.73	0.76	181	6.51	0.46
22	29.83	2.10	62	24.99	1.76	102	17.70	1.25	142	10.58	0.75	182	6.44	0.45
23	29.76	2.10	63	24.83	1.75	103	17.51	1.23	143	10.43	0.74	183	6.37	0.45
24	29.69	2.09	64	24.66	1.74	104	17.32	1.22	144	10.29	0.72	184	6.30	0.44
25	29.61	2.09	65	24.50	1.73	105	17.13	1.21	145	10.15	0.71	185	6.23	0.44
26	29.53	2.08	66	24.33	1.71	106	16.94	1.19	146	10.01	0.71	186	6.17	0.43
27	29.45	2.07	67	24.16	1.70	107	16.75	1.18	147	9.87	0.70	187	6.10	0.43
28	29.36	2.07	68	23.99	1.69	108	16.56	1.17	148	9.74	0.69	188	6.04	0.43
29	29.27	2.06	69	23.82	1.68	109	16.37	1.15	149	9.61	0.68	189	5.97	0.42
30	29.18	2.06	70	23.64	1.67	110	16.18	1.14	150	9.48	0.67	190	5.91	0.42
31	29.09	2.05	71	23.47	1.65	111	16.00	1.13	151	9.36	0.66	191	5.85	0.41
32	28.99	2.04	72	23.29	1.64	112	15.81	1.11	152	9.23	0.65	192	5.79	0.41
33	28.90	2.04	73	23.11	1.63	113	15.62	1.10	153	9.11	0.64	193	5.73	0.40
34	28.79	2.03	74	22.94	1.62	114	15.44	1.09	154	9.00	0.63	194	5.67	0.40
35	28.69	2.02	75	22.76	1.60	115	15.25	1.07	155	8.88	0.63	195	5.61	0.40
36	28.58	2.01	76	22.58	1.59	116	15.07	1.06	156	8.77	0.62	196	5.55	0.39
37	28.47	2.01	77	22.40	1.58	117	14.88	1.05	157	8.66	0.61	197	5.50	0.39
38	28.36	2.00	78	22.21	1.57	118	14.70	1.04	158	8.55	0.60	198	5.44	0.38
39	28.25	1.99	79	22.03	1.55	119	14.52	1.02	159	8.44	0.59	199	5.39	0.38
40	28.13	1.98	80	21.85	1.54	120	14.34	1.01	160	8.33	0.59	200	5.33	0.38

Nota**.** *Obtenido de "Diseño Estructural en Acero" (Zapata, L. 1967)*

Anexo B Tablas de resistencia de diseño de ángulos dobles 1 x 1

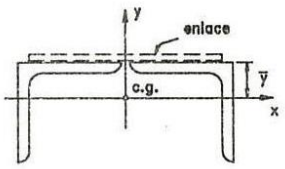


TABLA DE RESISTENCIAS DE DISEÑO

DE ANGULOS DOBLES EN COMPRESION AXIAL

$\phi_c P_n$ (toneladas)

A C E R O F_y = 2530 kg/cm²

TAMAÑO		1 x 1			Número de Enlaces
ESPESOR		1/4	3/16	1/8	
PESO (kg/m)		4.44	3.55	2.38	
Longitud Efectiva Eje X-X en cm.	0	12.2	9.4	6.5	
	30	11.2	8.7	6.0	
	60	8.6	6.8	4.8	
	90	5.0	4.5	3.2	
	120	3.3	2.7	1.9	
	150	***	1.7	1.2	
Longitud Efectiva Eje Y-Y en cm.	0	12.2	9.4	6.5	
	30	11.7	8.9	6.0	1
	60	11.1	8.4	5.4	
	90	10.4	7.9	5.0	2
	120	9.1	7.0	4.5	
	150	8.1	6.3	4.1	
	180	6.6	5.2	3.4	
	210	5.1	4.1	2.7	
	240	3.8	3.0	2.1	3
	270	2.9	2.3	1.6	
	300	2.3	1.8	1.3	
	330	2.2	1.8	1.2	
	360	1.8	1.5	1.0	4
	390	1.5	1.2	0.9	
AREA (cm^2)		5.645	4.387	3.026	
rx (cm)		0.737	0.754	0.772	
ry (cm)		1.833	1.889	1.948	
ȳ (cm)		0.861	0.808	0.752	
C_w (cm^6)		20	15	10	

Nota**.** *Obtenido de "Diseño Estructural en Acero" (Zapata, L. 1967)*

Anexo C Tablas de resistencia de diseño de ángulos dobles 1 ¼ x 1 ¼

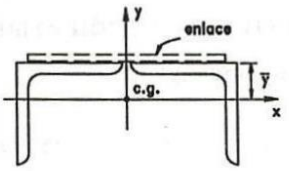


TABLA DE RESISTENCIAS DE DISEÑO

DE ANGULOS DOBLES EN COMPRESION AXIAL

$\phi_c P_n$ (toneladas)

A C E R O F_y = 2530 kg/cm²

TAMAÑO		1 1/4 x 1 1/4			Número de Enlaces
ESPESOR		1/4	3/16	1/8	
PESO (kg/m)		5.72	4.41	3.01	
Longitud Efectiva Eje X-X en cm.	0	15.6	12.1	8.2	
	30	14.8	11.5	7.9	
	60	12.6	9.8	6.8	
	90	9.7	7.6	5.3	
	120	6.7	5.3	3.8	
	150	4.3	3.5	2.5	
	180	3.0	2.4	1.7	
Longitud Efectiva Eje Y-Y en cm.	0	15.6	12.1	8.2	
	30	15.0	11.5	7.8	1
	60	14.4	10.7	6.9	
	90	13.5	10.0	6.2	
	120	12.9	9.6	5.8	2
	150	11.6	8.6	5.2	
	180	10.8	8.1	4.9	
	210	9.3	7.1	4.3	
	240	7.8	6.0	3.7	3
	270	6.3	4.9	3.1	
	300	5.0	3.9	2.5	
	330	4.7	3.7	2.4	
	360	3.9	3.0	2.0	
	390	3.3	2.6	1.7	4
	420	2.8	2.2	1.5	
	450	2.4	1.9	1.3	
	480	2.1	1.6	1.1	
AREA (cm^2)		7.290	5.594	3.832	
rx (cm)		0.937	0.998	0.978	
ry (cm)		2.347	2.406	2.465	
$\bar{y}$ (cm)		1.024	0.968	0.912	
C_w (cm^6)		60	45	30	

Nota. *Obtenido de "Diseño Estructural en Acero"* **(Zapata, L. 1967)**

Anexo D Tablas de resistencia de diseño de ángulos dobles 1 ½ x 1 ½

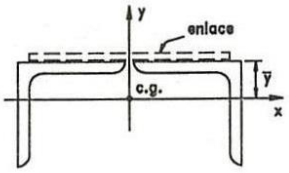


TABLA DE RESISTENCIAS DE DISEÑO

DE ANGULOS DOBLES EN COMPRESION AXIAL

$\phi_c P_n$ (toneladas)

A C E R O F_y = 2530 kg/cm²

TAMAÑO		1 1/2 x 1 1/2				Número de Enlaces
ESPESOR		1/4	3/16	5/32 *	1/8	
PESO (kg/m)		6.53	5.36	4.53	3.67	
Longitud Efectiva Eje X-X en cm.	0	19.1	14.6	12.3	10.0	
	30	18.4	14.1	11.9	9.6	
	60	16.6	12.7	10.8	8.7	
	90	13.8	10.7	9.1	7.4	
	120	10.8	8.4	7.2	5.8	
	150	7.8	6.2	5.3	4.3	
	180	5.4	4.3	3.7	3.1	
	210	4.0	3.2	2.7	2.2	
Longitud Efectiva Eje Y-Y en cm.	0	19.1	14.6	12.3	10.0	
	30	18.4	14.1	11.8	9.5	
	60	17.5	13.1	10.8	8.6	1
	90	16.8	12.2	9.9	7.6	
	120	15.5	11.3	9.0	6.7	
	150	15.2	10.9	8.6	6.3	
	180	13.9	10.1	7.9	5.7	2
	210	12.3	9.0	7.1	5.1	
	240	10.6	8.7	6.9	4.9	
	270	10.4	7.7	6.1	4.4	3
	300	8.9	6.7	5.3	3.9	
	330	7.5	5.6	4.5	3.3	
	360	6.2	4.7	3.8	3.2	
	390	5.2	3.9	3.3	2.8	
	420	4.4	3.4	2.8	2.5	
	450	3.8	3.4	2.8	2.2	4
	480	3.8	3.0	2.5	1.9	
	510	3.4	2.6	2.2	1.7	
	540	3.0	2.3	1.9	1.5	
AREA (cm^2)		8.903	6.774	5.73	4.639	
rx (cm)		1.140	1.161	1.17	1.181	
ry (cm)		2.863	2.923	2.94	2.984	
$\bar{y}$ (cm)		1.184	1.128	1.10	1.069	
C_w (cm^6)		149	112	93	72	

* no común en el mercado

Nota**.** *Obtenido de "Diseño Estructural en Acero" (Zapata, L. 1967)*

Anexo E Tablas de resistencia de diseño de ángulos dobles 2 x 2

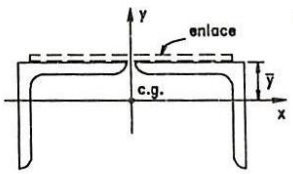


TABLA DE RESISTENCIAS DE DISEÑO

DE ANGULOS DOBLES EN COMPRESION AXIAL

$\phi_c P_n$ (toneladas)

A C E R O F_y = 2530 kg/cm²

TAMAÑO		2 x 2					Número de Enlaces
ESPESOR		3/8	5/16	1/4	3/16	1/8	
PESO (kg/m)		14.01	11.68	9.51	7.27	4.92	
Longitud Efectiva Eje X-X en cm.	0	37.8	31.9	26.0	19.9	12.2	
	60	34.8	29.5	24.1	18.4	11.4	
	90	31.4	26.7	21.9	16.7	10.5	
	120	27.2	23.2	19.1	14.7	9.4	
	150	22.6	19.4	16.0	12.4	8.0	
	180	18.1	15.5	12.9	10.0	6.7	
	210	13.8	12.0	10.0	7.8	5.4	
	240	10.6	9.2	7.7	6.0	4.2	
	270	8.4	7.2	6.1	4.7	3.3	
	300	6.8	5.9	4.9	3.8	2.7	
Longitud Efectiva Eje Y-Y en cm.	0	37.5	31.9	26.0	19.9	12.2	
	60	35.7	29.9	24.1	15.2	11.2	
	90	34.7	28.8	22.8	16.8	10.1	
	120	33.7	27.7	21.7	15.6	9.1	1
	150	31.9	26.3	20.5	14.4	8.1	
	180	29.4	24.3	19.0	13.2	7.1	
	210	29.9	24.7	17.0	11.8	6.3	
	240	27.8	23.1	17.9	10.4	5.4	
	270	25.4	21.1	16.4	11.2	5.5	2
	300	25.1	20.9	16.4	10.3	5.0	
	330	23.0	19.2	15.1	9.2	4.9	
	360	20.7	17.4	13.7	9.4	4.5	3
	390	19.9	16.8	12.3	8.5	4.1	
	420	17.9	15.1	12.0	8.3	4.0	
	450	15.9	13.5	10.8	7.5	3.7	4
	480	14.0	11.9	9.5	6.7	3.5	
AREA (cm^2)		17.548	14.839	12.124	9.226	6.194	
rx (cm)		1.509	1.527	1.547	1.567	1.590	
ry (cm)		3.779	3.637	3.697	3.958	4.021	
$\bar{y}$ (cm)		1.615	1.520	1.504	1.445	1.387	
C_w (cm^6)		940	783	627	470	313	

Nota. *Obtenido de "Diseño Estructural en Acero" (Zapata, L. 1967)*

Anexo F Tablas de resistencia de diseño de ángulos dobles 2 ½ x 2 ½

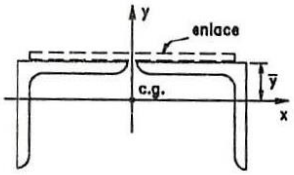


TABLA DE RESISTENCIAS DE DISEÑO

DE ANGULOS DOBLES EN COMPRESION AXIAL

$\phi_c P_n$ (toneladas)

A C E R O F_y = 2530 kg/cm²

TAMAÑO		2 1/2 x 2 1/2				Número de Enlaces
ESPESOR		3/8	5/16	1/4	3/16	
PESO (kg/m)		17.58	14.90	12.22	9.15	
Longitud Efectiva Eje X-X en cm.	0	48.0	40.5	33.0	25.1	
	60	45.6	38.6	31.5	23.9	
	90	42.8	36.2	29.6	22.6	
	120	39.2	33.2	27.2	20.8	
	150	34.9	29.7	24.3	18.7	
	180	30.4	25.9	21.3	16.5	
	210	25.7	22.0	18.2	14.1	
	240	21.3	18.3	15.1	11.9	
	270	17.1	14.7	12.3	9.7	
	300	13.8	11.9	9.9	7.9	
	330	11.4	9.9	8.2	6.5	
	360	9.6	8.3	6.9	5.5	
Longitud Efectiva Eje Y-Y en cm.	0	48.0	40.5	33.0	25.1	
	60	45.7	38.4	31.1	23.5	
	90	44.1	36.7	29.4	22.0	
	120	42.8	35.2	27.8	20.3	1
	150	41.5	33.8	26.4	18.8	
	180	39.8	32.3	25.1	17.3	
	210	37.6	30.6	24.1	15.8	
	240	34.9	28.4	23.1	14.5	
	270	36.4	29.4	22.0	13.1	
	300	34.5	27.9	20.9	13.4	
	330	32.2	26.2	19.6	12.5	
	360	29.7	24.3	18.3	11.7	2
	390	30.3	22.3	16.9	10.8	
	420	28.2	23.1	17.4	9.9	
	450	26.0	21.4	16.2	9.0	
	480	23.8	19.6	14.9	9.4	
	510	21.6	17.9	13.7	8.7	3
	540	21.7	16.1	12.4	8.1	
	570	19.8	16.4	12.5	8.1	4
	600	17.8	14.8	11.4	7.6	
AREA (cm^2)		22.387	18.903	15.355	11.613	
rx (cm)		1.913	1.933	1.953	1.976	
ry (cm)		4.811	4.870	4.932	4.995	
$\bar{y}$ (cm)		1.935	1.880	1.821	1.763	
C_w (cm^6)		2868	2390	1912	1434	

Nota. *Obtenido de "Diseño Estructural en Acero" (Zapata, L. 1967)*

Anexo G Tablas de resistencia de diseño de ángulos dobles 3 x 3

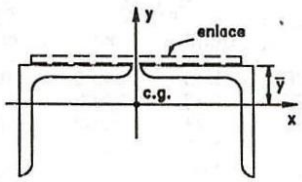


TABLA DE RESISTENCIAS DE DISEÑO

DE ANGULOS DOBLES EN COMPRESION AXIAL

$\phi_c P_n$ (toneladas)

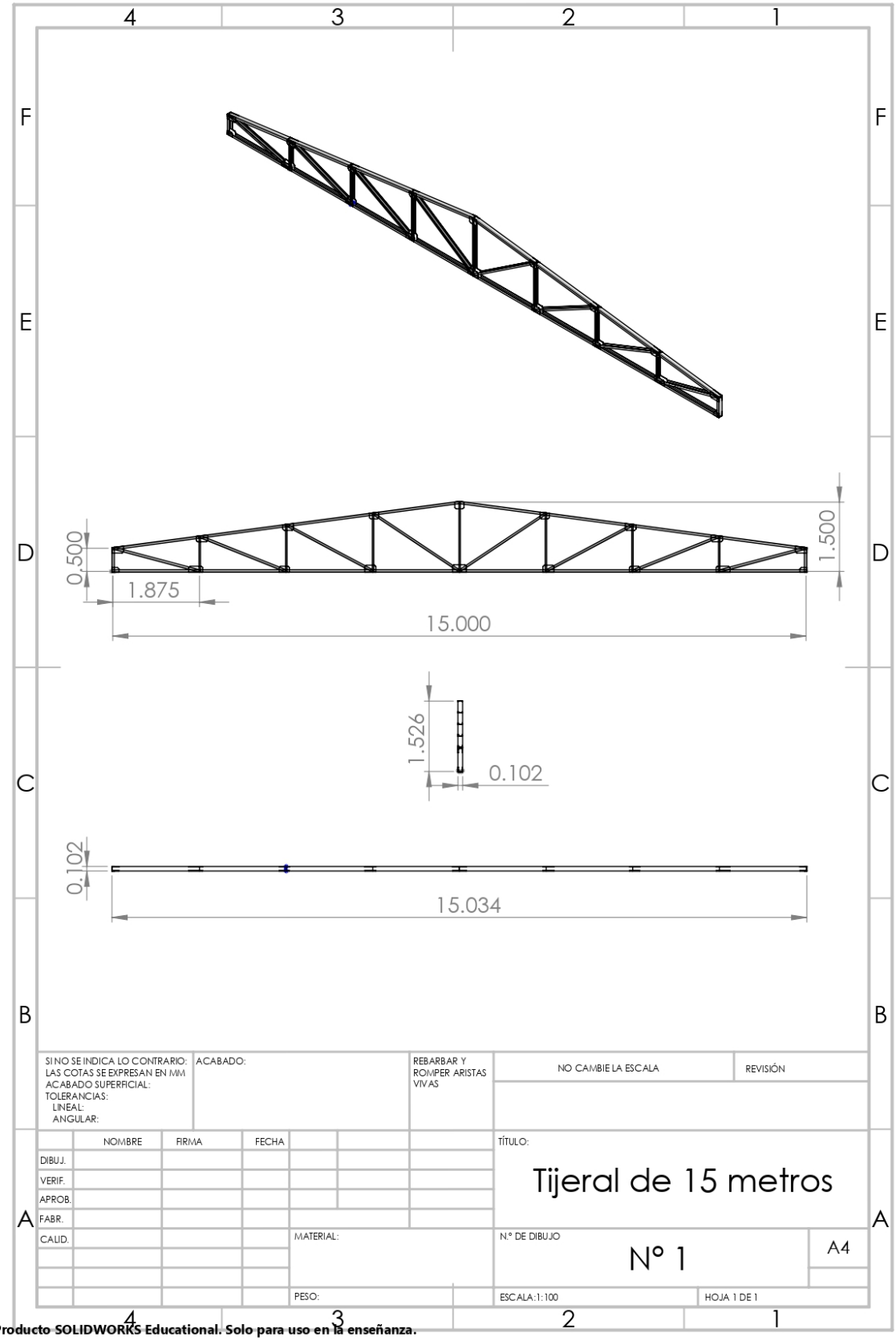
A C E R O F_y = 2530 kg/cm²

TAMAÑO		3 x 3				Número de Enlaces
ESPESOR		1/2	3/8	5/16	1/4	
PESO (kg/m)		28.01	21.43	18.16	14.60	
Longitud Efectiva Eje X-X en cm.	0	76.3	58.6	49.5	40.0	
	60	73.6	56.6	47.8	38.7	
	120	66.1	51.0	43.1	35.0	
	150	61.0	47.2	40.0	32.4	
	180	55.3	42.9	36.4	29.6	
	210	49.2	38.3	32.6	26.6	
	240	43.0	33.6	28.7	23.4	
	270	37.0	29.0	24.8	20.3	
	300	31.2	24.6	21.1	17.4	
	330	25.9	20.5	17.7	14.5	
	360	21.7	17.2	14.8	12.2	
	390	18.5	14.7	12.6	10.4	
	420	16.0	12.7	11.0	9.0	
	450	13.9	11.0	9.5	7.8	
Longitud Efectiva Eje Y-Y en cm.	0	76.3	58.6	49.5	40.0	
	60	73.3	56.3	47.4	38.2	
	120	70.0	52.3	43.3	34.4	1
	180	67.0	48.9	39.7	30.5	
	240	62.1	45.1	36.1	27.0	
	300	60.9	43.8	34.6	25.1	
	360	55.6	40.3	31.8	22.8	2
	420	49.2	36.0	28.6	20.5	
	480	47.6	31.1	24.9	17.9	
	540	41.7	30.8	24.6	17.6	
	600	35.7	26.6	21.4	15.4	3
	660	33.6	22.5	18.1	13.3	
	690	31.0	20.5	16.6	12.3	
AREA (cm^2)		35.484	27.226	22.903	18.581	
rx (cm)		2.281	2.319	2.342	2.362	
ry (cm)		5.727	5.844	5.907	5.969	
$\bar{y}$ (cm)		2.367	2.256	2.197	2.139	
C_w (cm^6)		9516	7137	5948	4758	

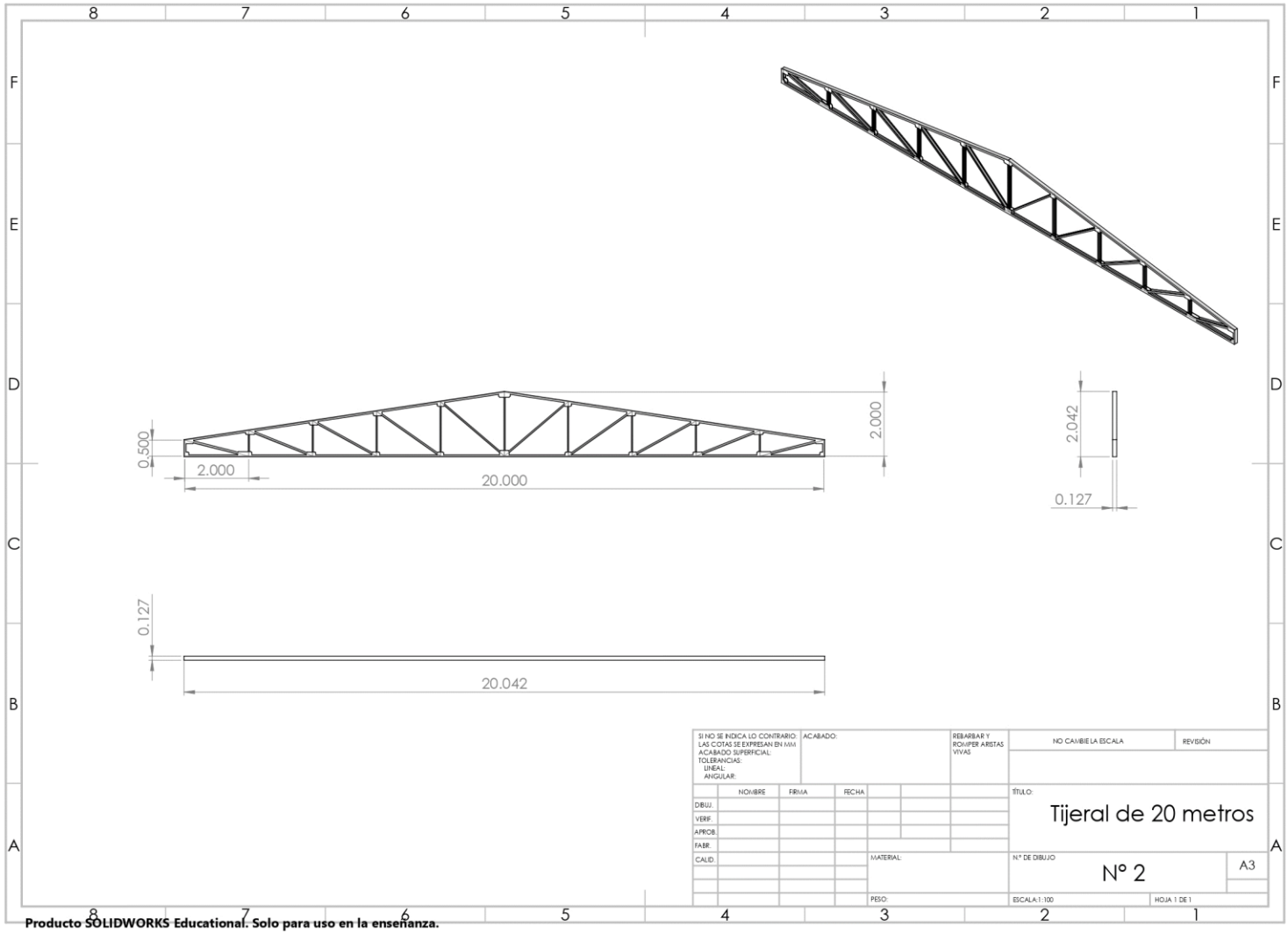
Nota. *Obtenido de "Diseño Estructural en Acero" (Zapata, L. 1967)*

Planos

Plano A Tijeral de 15 metros



Plano B Tijeral 20 metros



Plano C Tijeral de 27 metros