



Universidad
Central

INDEPENDENCIA - PLURALISMO - COMPROMISO

**UNIVERSIDAD CENTRAL DE CHILE
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERIA**

**SOFTWARE DE PREDISEÑO DE MARCOS ESPECIALES RESISTENTES
A MOMENTO DE HORMIGÓN ARMADO**

PATRICIO NICOLÁS CARRASCO VINAGRE

ALEJANDRO ESTEBAN PÉREZ HORTA

Profesor Guía:

MARIO ALFREDO PINTO MAIRA

Profesores Informantes:

PABLO ALCAÍNO REYES

DANIELA BRIZUELA VALENZUELA

SANTIAGO, CHILE, 2021.



Universidad
Central

INDEPENDENCIA - PLURALISMO - COMPROMISO

**UNIVERSIDAD CENTRAL DE CHILE
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA**

**SOFTWARE DE PREDISEÑO DE MARCOS ESPECIALES RESISTENTES
A MOMENTO DE HORMIGÓN ARMADO**

PATRICIO NICOLÁS CARRASCO VINAGRE

ALEJANDRO ESTEBAN PÉREZ HORTA

Memoria para optar al Título
de Ingeniero Civil en Obras Civiles.

Profesor Guía:

MARIO ALFREDO PINTO MAIRA

Profesores Informantes:

PABLO ALCAÍNO REYES

DANIELA BRIZUELA VALENZUELA

SANTIAGO, CHILE, 2021.



Universidad
Central

INDEPENDENCIA - PLURALISMO - COMPROMISO

**UNIVERSIDAD CENTRAL DE CHILE
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA**

**SOFTWARE DE PREDISEÑO DE MARCOS ESPECIALES RESISTENTES
A MOMENTO DE HORMIGÓN ARMADO**

Memoria preparada bajo la supervisión de
la comisión integrada por los profesores:

MARIO ALFREDO PINTO MAIRA

PABLO ALCAÍNO REYES

DANIELA BRIZUELA VALENZUELA

Quienes recomiendan que sea aceptada
para completar las exigencias del Título
de Ingeniero Civil en Obras Civiles.

SANTIAGO, CHILE, 2021.

Dedicatoria

*A mis padres, abuelos, tíos, primos, amigos,
compañeros y profesores*

APH

Agradecimientos

Primero que todo a mis padres, quienes me criaron y creyeron siempre en mí, a mis tíos y primos quienes me brindaron su ayuda y ánimo en incontables veces, a mis abuelos, recordando con especial cariño a mi Lala (no le gustaba que le dijeran abuela), quien poseía una sabiduría infinita y siempre fue la fan número uno de sus hijos y nietos, a mis amigos por llenarme de alegría, destacando a Carlos Gallardo, quien me regaló la tarjeta madre y procesador para este fin, a mis compañeros de escuela, en especial a Paulina Donoso, quien siempre me ha aconsejado en el transcurso de mi carrera, a mis compañeros de informática, Rubén Contreras y Sebastián Hanania, por resolverme algunas dudas, también a mi compañero de proyecto de título Patricio Carrasco, con quien aprendí el lenguaje de programación que se utilizó en el proyecto, y a todos mis profesores, en especial al profesor Mario Pinto, quien nos guió en esta memoria con mucho entusiasmo de principio a fin y nos proporcionó gran parte del material, a los profesores Hernán Olmí y Nelson Sepúlveda, por incentivar me a programar e innovar, y, al profesor Alejandro Torres, quien me convenció de no abandonar los estudios y cambiarme de ingeniería en construcción a ingeniería civil en obras civiles.

Alejandro Pérez Horta

Resumen

El prediseño de marcos especiales resistentes a momento de hormigón armado es un proceso iterativo que involucra diversos criterios de diseño enfocados en la seguridad de la estructura y sus costos. Esto requiere de herramientas computacionales debido a la gran cantidad de operaciones que involucra, tales como análisis matricial de estructuras, diseño sismorresistente, diseño estructural, detallamiento, y cubicación. Este proyecto es la memoria del desarrollo de un *software*, *β -beam*, el que calcula y diseña marcos planos resistentes a momento, enfocándose en un prediseño económico, para entregar factores de utilización de esfuerzos internos, deformaciones, detallamiento y cubicación de sus elementos, bajo los criterios del código ACI 318S-14 y la norma NCh 433. Toma como referencia las características de programas conocidos dentro de la industria, tales como RISA-2D/3D y SAP2000. En esta memoria se presentan los criterios de diseño utilizados habitualmente en Chile, respaldados por códigos y normas, incorporando métodos para minimizar los costos para elegir la mejor combinación de materiales de acuerdo con las limitaciones del software. Este software se ajusta tanto al entorno laboral como académico, debido a su fácil uso y licencia no restrictiva, siendo así una opción más dentro de la categoría de *software* libre, quedando a disposición del lector el repositorio de este, junto a archivos de ejemplo para demostrar sus características.

Palabras clave: ACI 318S-14, NCh 433, prediseño, hormigón armado, marcos, *software*, *Python*, desarrollo, *Open Source*, WTFPL.

Abstract

The pre-design of special reinforced concrete moment resisting frames is an iterative process that involves several design criteria focused on the safety of the structure and its costs. This requires computational tools due to the large number of operations involved, such as matrix analysis of structures, seismic-resistant design, structural design, detailing, and quantification. This project is the report of the development of a software, β -beam, which calculates and designs flat frames resistant to moment, focusing on an economic pre-design, to deliver utilization factors of internal forces, deformations, detailing and quantification of its elements, under the criteria of the ACI 318S-14 code and the NCh 433 standard. It takes as a reference the characteristics of well-known programs within the industry, such as RISA-2D/3D and SAP2000. This report presents the design criteria commonly used in Chile, supported by codes and standards, incorporating methods to minimize costs to choose the best combination of materials according to the limitations of the software. This software is suitable for both work and academic environments, due to its ease of use and non-restrictive license, being one more option within the category of free software and leaving at the reader's disposal its repository, along with example files to demonstrate its features.

Keywords: ACI 318S-14, NCh 433, pre-design, reinforced concrete, SMF, software, Python, development, Open Source, WTFPL.

Índice

Resumen	i
Abstract.....	ii
Índice	iii
Lista de imágenes	vi
Lista de tablas	viii
1. Introducción.....	1
1.1. Presentación del proyecto	1
1.2. Motivación	2
2. Objetivos.....	4
2.1. Objetivo general.....	4
2.2. Objetivos específicos	4
2.3. Resultados esperados	4
3. Alcances y limitaciones	5
3.1. Alcances	5
3.2. Limitaciones.....	5
4. Metodología.....	7
4.1. Búsqueda de bibliografía	7
4.2. Herramientas de desarrollo, entorno de trabajo	7
4.3. Revisión de programas existentes.....	8
4.4. Planificación del flujo del programa y su desarrollo	8
4.5. Comprobación de resultados.....	8
5. Estado del arte	9
6. Cálculo de esfuerzos internos	12
6.1. Diseño sismo resistente.....	12
6.1.1. Método estático	12
6.1.2. Período fundamental de la estructura	14
6.2. Análisis matricial de estructuras	15
6.2.1. Grados de libertad.....	15
6.2.2. Cargas Tributarias.....	16
6.2.3. Matriz de rigidez.....	19
6.2.4. Tramos rígidos.....	21
7. Criterios de diseño	24
7.1. Consideraciones de los materiales	24
7.1.1. Consideraciones del hormigón	24
7.1.2. Consideraciones del acero	24
7.2. Método de diseño por resistencia.....	25

7.2.1.	Principios básicos de diseño	25
7.2.2.	Combinaciones de carga	25
7.2.3.	Factor de reducción de resistencia.....	26
7.3.	Diseño a esfuerzos axiales, de flexión y flexo-compresión.....	26
7.3.1.	Generalidades	26
7.3.2.	Resistencia axial a la compresión máxima	26
7.3.3.	Resistencia axial a tracción máxima.....	26
7.4.	Diseño al corte	27
7.4.1.	La resistencia al corte de diseño	27
7.4.2.	La resistencia nominal	27
7.4.3.	La cuantía mínima para refuerzo transversal en vigas	27
7.4.4.	Espaciamiento máximo de estribos en vigas	27
7.4.5.	Espaciamiento máximo de estribos en columnas	27
7.5.	Diseño de marcos especiales resistentes a momento	28
7.5.1.	Vigas.....	28
7.5.2.	Columnas	28
7.6.	Longitudes de desarrollo y empalmes.....	29
7.6.1.	Longitud de desarrollo para barras y alambres corrugados a tracción	29
7.6.2.	Longitud de empalmes por traslapo de barras corrugadas a tracción.....	29
7.6.3.	Ganchos estándar para el desarrollo de barras corrugadas a tracción	29
7.6.4.	Longitud de desarrollo para barras corrugadas a compresión	30
7.6.5.	Longitud de empalmes por traslapo de barras corrugadas a compresión	30
8.	Resultados.....	31
8.1.	Diagrama de flujo de la aplicación	32
8.2.	Interfaz y espacio de trabajo	33
8.3.	Datos de entrada.....	34
8.4.	Armado del marco	37
8.5.	Casos de cargas básicas	38
8.6.	Análisis sísmico	40
8.7.	Verificación de drift.....	41
8.8.	Diseño estructural de marcos especiales resistentes a momento	43
8.8.1.	Mapeo de datos	45
8.8.2.	Cálculo de vigas óptimas.....	48
8.8.3.	Corrección de momentos	55
8.8.4.	Cálculo de columnas óptimas	56
8.8.5.	Función de costos de toda la estructura	63
9.	Conclusiones y Recomendaciones	64
	Referencias	66

Bibliografía.....	71
Anexo A – Repositorios	72
A.1. Repositorio GitHub.....	72
A.2. Repositorio PyPi	72
Anexo B – Demostración	73
B.0. Enunciado del proyecto	73
B.1. Configuración del proyecto	73
B.2. Comprobación de esfuerzos	75
B.3. Verificación del drift máximo.....	83
B.4. Diseño de vigas y columnas.....	85
B.5. Detallamiento final	89
B.6. Curvas de interacción de columnas tipo por piso	104
Anexo C – Cálculo de expresiones algebraicas.....	112
C.1. Flexión simple.....	112
C.1.1. Refuerzo de tracción.....	112
C.1.2. Refuerzo de compresión	115
C.2. Flexo-compresión	117
C.2.1. Cálculo de cargas y momentos nominales.....	117
C.2.2. Curvas de interacción	120
C.3. Factor de corrección de área	121
Anexo D – Código Fuente β -beam	124

Lista de imágenes

Figura 1: Esquema metodología.	7
Figura 2: Transformación a un sistema equivalente con cargas puntuales.....	16
Figura 3: Empotramiento perfecto, carga distribuida constante.....	17
Figura 4: Empotramiento perfecto, carga distribuida constante.....	18
Figura 5: Penetración de elementos en nudos rígidos.	22
Figura 6: Barra con conexiones infinitamente rígidas.....	22
Figura 7: Simplificación de la curva esfuerzo deformación del acero (Park y Paulay, 1983).	24
Figura 8: Diagrama de flujo del programa.	32
Figura 9: Espacio de trabajo, β -beam.	33
Figura 10: Ejecución del programa, β -beam.	34
Figura 11: Input de usuario.....	35
Figura 12: Esquema de marco en 2D.....	37
Figura 13: Esquema de marco en 3D.....	38
Figura 14: Ángulos para áreas tributarias.	39
Figura 15: Casos de áreas tributarias.	39
Figura 16: Parámetros área trapezoidal.	40
Figura 17: Drifts por nivel.	42
Figura 18: Diagrama de flujo más detallado desde los resultados del cálculo estructural. ..	43
Figura 19: Zonas donde se debe cumplir con las cuantías mínimas.....	48
Figura 20: Corte transversal de una viga.	49
Figura 21: Envolvente del corte ampliando zona de rótula plástica.....	50
Figura 22: Ejemplo de estribos entrelazados.....	51
Figura 23: Detalle de viga en marco plano.....	53
Figura 24: Criterio de sumatoria de momentos en un nodo para un diseño columna fuerte y viga débil.	56
Figura 25: Curvas de interacción de una columna sometida a flexo-compresión.....	58
Figura 26: Detalle de columna en marco plano.....	60
Figura 27: Estribos interiores y exteriores.....	61
Figura 28: Datos de entrada, marco de prueba.	74
Figura 29: Modelo de marco inicial, RISA-3D.	75
Figura 30: Cargas muertas totales, RISA-3D.	76
Figura 31: Cargas vivas totales, RISA-3D.	76
Figura 32: Fuerzas sísmicas, RISA-3D.	77
Figura 33: Combinación 1.2 D + 1.4 E + L, RISA-3D.	77
Figura 34: Esfuerzos internos, RISA-3D.....	78

Figura 35: Esfuerzos internos, β -beam.	78
Figura 36: Reporte miembro 2, columna, RISA-3D.	79
Figura 37: Reporte miembro 2, columna, β -beam.	80
Figura 38: Reporte miembro 18, viga, RISA-3D.	81
Figura 39: Reporte miembro 18, viga, β -beam.	82
Figura 40: Desplazamientos mínimos y máximos por nivel, β -beam.	83
Figura 41: Primeras dimensiones sugeridas, β -beam.	84
Figura 42: Verificación drift, tercera vuelta.	85
Figura 43: Curva de interacción de la columna del tipo 1 del piso 1.	104
Figura 44: Curva de interacción de la columna del tipo 1 del piso 2.	105
Figura 45: Curva de interacción de la columna del tipo 1 del piso 3.	106
Figura 46: Curva de interacción de la columna del tipo 1 del piso 4.	107
Figura 47: Curva de interacción de la columna del tipo 2 del piso 1.	108
Figura 48: Curva de interacción de la columna del tipo 2 del piso 2.	109
Figura 49: Curva de interacción de la columna del tipo 2 del piso 3.	110
Figura 50: Curva de interacción de la columna del tipo 1 del piso 1.	111
Figura 51: Simplificación de la distribución de esfuerzos en una sección reforzada a tracción.	112
Figura 52: Distribución de esfuerzos en una sección doblemente reforzada.	115
Figura 53: Sección transversal de un posible elemento de hormigón armado.	118
Figura 54: Zonas del diagrama de interacción en una columna.	120

Lista de tablas

Tabla 7.1: Combinaciones de carga del software de acuerdo a la norma NCh 3171 (2010).	25
Tabla 8.1: Extracto de tabla de resultados de esfuerzos internos de los elementos.	45
Tabla B.1: Medidas de las secciones en la Iteración 1.	86
Tabla B.2: Costos y drift de la iteración 1.	86
Tabla B.3: Medidas de las secciones en la iteración 2.	87
Tabla B.4: Costos y drift de la iteración 2.	87
Tabla B.5: Medidas de las secciones en la iteración 2.	88
Tabla B.6: Costos y drift de la iteración 3.	88

1. Introducción

1.1. Presentación del proyecto

El proyecto consiste en la planificación, elaboración y publicación de un software open source (programa de código abierto), con una licencia no restrictiva para que pueda desempeñarse tanto en entornos académicos como laborales.

Este software, denominado *β -beam*, desarrolla el prediseño de marcos especiales resistentes a momento, calculando las secciones de los elementos utilizados en función de su costo, y genera un informe en base a las características calculadas. *β -beam* busca replicar características esenciales encontradas en otros programas comúnmente utilizados para el análisis y diseño estructural. Para esto, se calculan los esfuerzos internos por medio del método matricial de la rigidez, diseñando vigas y columnas con los criterios de normativas y códigos de diseño aprobados para el desarrollo de proyectos de ingeniería civil aplicables en Chile, tales como la norma chilena “NCh 433 Of. 96 Mod. 2012” (NCh 433, 1996), verificando la respuesta de la estructura ante fuerzas producidas por un sismo por medio del método estático; el Código ACI 318S-14 (Comité ACI 318, 2014), para el diseño de las vigas y columnas; y buenas prácticas para garantizar un prediseño conservador. Todo lo anterior se lleva a cabo pidiendo al usuario pocos datos de entrada, tales como los atributos necesarios para la identificación de la geometría de un marco regular plano, atributos de los materiales a utilizar y las características del tipo de zona, suelo y edificación para la estructura a proyectar, generando como resultado un informe con los datos de esfuerzos internos, desplazamientos en los nodos, curvas de interacción, cubicación, esquemas y gráficos que sustentan la información, resumiendo todo lo referente al proyecto. El programa se desarrolla utilizando principalmente el lenguaje de programación *Python* y se encuentra publicado en las plataformas *GitHub* y *PyPi* para su libre distribución y uso.

1.2. Motivación

Dentro del dominio de la ingeniería civil en obras civiles, existen softwares que permiten facilitar el flujo de trabajo; variando desde herramientas que unen diferentes disciplinas, como ingeniería, arquitectura y construcción, denominadas bajo el nombre de tecnologías BIM, *Building Information Modeling* (Autodesk, 2021); hasta herramientas de un área específica como diseño estructural, en donde se encuentran softwares como RISA-2D/3D y SAP2000. Mientras que las soluciones son variadas, no terminan de satisfacer un mercado complejo, como lo son los proyectos de obras civiles, donde es común que los ingenieros deban hacer programas nuevos para cada particularidad que surja dentro de un proyecto y así complementar con el conjunto de softwares que dispongan. Adicionalmente, existe la permanente necesidad de abaratar costos, pues es común que los softwares actuales se consideren muy caros (Araya-Castillo y Pedreros-Gajardo, 2013) y la incorporación de ellos dentro del ambiente académico se ve mermada ya que no todos cuentan con una versión estudiantil, dificultando la adopción de estas tecnologías en las nuevas generaciones y, junto con la situación mundial debido a la pandemia al momento de publicar de este documento, no es de extrañar que surja la necesidad de instalar software de forma ilícita por parte de los alumnos cuando en tiempos normales ya era una práctica común (Gupta y Jamwal, 2012).

Se sabe que el uso de software presenta beneficios indiscutibles y cada vez se demandan más (Bridgwater, 2020; Grand Canyon University, 2020), por lo que entender cómo funcionan internamente es un gran beneficio. Sin embargo, durante el 2018, se estimó que en Chile un 55% del software utilizado por los usuarios era pirata, lo que genera un potencial riesgo para la información personal debido a la relación directa de estas prácticas con la presencia de *malware* (Business Software Alliance [BSA], 2018), adicionalmente, los usuarios se arriesgan a recibir multas que pueden alcanzar cifras de 40 UTM (Poder Judicial de Chile, s.f., como se citó en Microjuris, 2019). Esto se suma al hecho de que el *software* es percibido como un producto caro (Araya-Castillo y Pedreros-Gajardo, 2013) y su uso sin licencias válidas no es percibido por la gente como un acto inmoral (Kini, s.f.), además de que muchas

veces la gente no es consciente o capaz de comprobar que el software que usa cuente con todos los permisos pertinentes (Cooperativa, 2016; Vallejos, s.f.).

Por otro lado, una misma solución se puede generar con diferentes métodos, y cada método se puede implementar en una cantidad indeterminada de formas, que es la premisa con la que las empresas trabajan para seguir desarrollando alternativas y dar más soluciones a problemas que no son nuevos o que ya han sido solucionados de otra manera (Bridgwater, 2020; Grand Canyon University, 2020) y, así como en un momento saber inglés era una cualidad adicional a la hora de buscar trabajo y hoy en día es un requisito mínimo para muchas postulaciones, lo mismo está pasando con la programación que, impulsada por el avance de la tecnología, ha logrado que la carrera de desarrollador de aplicaciones sea la más demandada a nivel mundial, compitiendo en países desarrollados con carreras del área de la salud y la electrónica (Drummond, 2017; Rodríguez, 2019; Fanjul, 2019).

Finalmente, la programación es compatible con todos los campos del conocimiento, proponiendo soluciones revolucionarias que se han hecho notar en el último tiempo en química (Nieves, 2020), termodinámica (Hao, 2020) y biomedicina (Heaven, 2020), lo que despierta la necesidad de aprovechar nuevas habilidades que empiezan a ser comunes a todas las carreras de ingeniería que han adoptado ramos de programación.

Bajo todas estas premisas, se propuso en este proyecto investigar sobre el desarrollo de aplicaciones, buscando la integración de conocimientos de ingeniería civil e informática, a la vez que se profundizan materias pasadas, permitiendo explorar tempranamente algunos temas novedosos de especialización a la vez que se intenta aportar en la adopción de nuevas alternativas que, en el corto plazo, puede beneficiar enormemente al sector académico al tratarse de una herramienta desarrollada con el fin de que otros la puedan usar libremente, teniendo siempre las fuentes oficiales para adquirirla, eliminando así la necesidad de buscar el programa en sitios de terceros, lo que indirectamente disminuye los riesgos por malware disfrazado entre tales enlaces y se traduce en beneficios económicos.

2. Objetivos

2.1. Objetivo general

Desarrollar un software libre y de código abierto para el prediseño óptimo de marcos especiales resistentes a momento de hormigón armado que pueda ser utilizado como herramienta de apoyo y desarrollo tanto en el área académica como laboral.

2.2. Objetivos específicos

- Identificar las expresiones de diseño y buenas prácticas requeridas para el cálculo de elementos de hormigón armado en marcos especiales resistentes a momento, a partir del código ACI 318S-14 (Comité ACI 318, 2014), la NCh 433 (1996) y las distintas fuentes bibliográficas.
- Crear un software simple orientado al diseño de marcos especiales resistentes a momento con un enfoque en el prediseño económico.
- Disponer del software en una plataforma de distribución conocida para su fácil descarga.
- Obtener resultados de los esfuerzos de una estructura de ejemplo modelada en β -beam y en RISA-2D/3D con una aproximación razonable.

2.3. Resultados esperados

- Desarrollar un software libre y de código abierto que permita el prediseño económico de marcos especiales resistentes a momento.
- Profundizar conocimientos relacionados al análisis matricial de estructuras, hormigón armado y programación orientada al desarrollo de software de ingeniería.
- Hacer coincidir los resultados obtenidos de β -beam respecto de los obtenidos de los programas.

3. Alcances y limitaciones

3.1. Alcances

Se abordan solo los tópicos necesarios para el desarrollo del software, el cual solo realiza el prediseño económico de marcos especiales resistentes a momento en dos dimensiones y se limita a diseños destinados a edificación de viviendas bajo los criterios de diseño del código ACI 318S-14 (Comité ACI 318, 2014) y el método estático de la norma NCh 433 (1996) para el diseño sismo resistente.

3.2. Limitaciones

- El criterio de diseño de marcos especiales resistentes a momento se realiza de acuerdo con el capítulo 18 del código ACI 318S-14 (Comité ACI 318, 2014).
- Limitaciones según se indican para el uso del método estático, siendo estas las establecidas por la norma NCh 433 (1996), apartado 6.2.1, donde se define que tipo de estructuras son válidas para aplicar el método.
- Los marcos se consideran empotrados en las bases, se desprecian los efectos de esbeltez y los aumentos en los esfuerzos producto de una torsión accidental.
- Al ser marcos planos, el análisis de flexo-compresión es en una sola dirección.
- Las dimensiones de ancho y alto de las vigas y columnas varían entre 30 y 90 centímetros, siendo las columnas de forma cuadrada.
- Los diámetros de barras longitudinales principales varían de 16 mm a 36 mm, tanto en vigas como columnas, para refuerzo lateral en vigas desde 8 mm, y los estribos, de 10 mm a 12 mm para todos los casos.

- La resistencia a compresión del hormigón (f'_c) debe estar entre 21 MPa y 70 MPa, y la tensión de fluencia del acero (f_y) a utilizar es 420 MPa.
- La estructura debe tener por lo menos 2 vanos, 2 pisos por marco y se debe componer de al menos 3 marcos.
- Los nodos de un marco coinciden con el centro de gravedad de la sección de cada elemento de la estructura detallada.
- Se desprecian los efectos de la amplificación de las fuerzas horizontales producto de una torsión accidental.
- Se verifica el marco interior más desfavorable (con más carga tributaria).
- No se verifican los nodos en los pórticos por tratarse de un prediseño

4. Metodología

La metodología se resume en el siguiente esquema:



Figura 1: Esquema metodología.

4.1. Búsqueda de bibliografía

Se recopilaron todos los documentos necesarios para limitar el desarrollo del programa y los procedimientos de diseño estructural ajustados para un proyecto aplicable en Chile, esto es, normativas vigentes, códigos y manuales aprobados y pertinentes para el desarrollo de proyectos dentro del país.

4.2. Herramientas de desarrollo, entorno de trabajo

Se buscaron todas las herramientas y tecnologías mínimas necesarias para llevar a cabo el desarrollo y posterior publicación y distribución de un software que cumple con los estándares de softwares open source, lo que involucra licencias, lenguajes de programación, entornos de programación y sistema operativo de trabajo.

4.3. Revisión de programas existentes

Se buscaron programas semejantes que ya han sido usados en proyectos de ingeniería en Chile para enfocar el desarrollo del programa, establecer las características mínimas que se deben desarrollar y establecer un flujo ordenado que represente de manera precisa el comportamiento interno del programa. También se buscaron otros programas similares para comprobar el estado actual de desarrollo de alternativas que no tengan licencias restrictivas y que favorezcan la libre distribución y uso de estos en entornos académicos y laborales.

4.4. Planificación del flujo del programa y su desarrollo

Se detectaron las características básicas para implementar en el desarrollo de un programa que permita realizar el análisis estructural de un modelo matemático de marcos especiales resistentes a momento, teniendo en cuenta criterios de diseño sismorresistentes, y automatizarlo para que realice un prediseño en función del precio unitario de los materiales utilizados y la resistencia de los elementos de una estructura dada, todo lo anterior aplicable a la realidad de un proyecto para llevar a cabo en Chile, bajo las respectivas normativas, códigos y manuales aprobados y suficientes para proyectos de ingeniería civil.

4.5. Comprobación de resultados

Se comprobaron los resultados medibles en otros programas, siendo estos los que competen al análisis estructural, para corroborar que los datos previos al diseño fuesen exactos. Esta comprobación se llevó a cabo por medio del programa RISA-2D/3D, mientras que se omiten desarrollos respecto al diseño para directamente presentar los resultados al no haber un programa con criterios semejantes.

5. Estado del arte

Los softwares de análisis y diseño estructural son útiles para los ingenieros al permitir ahorrar una gran cantidad tiempo facilitando diferentes actividades tales como es el automatizar la elaboración de un modelo matemático que describa una estructura o su posterior cálculo de esfuerzos internos. Estos cálculos suelen ser complejos y extensos, al punto de que no es rentable ni seguro para un individuo realizarlos sin ayuda de computadores que lo hagan de forma automática mediante un software especializado, ya que para el cálculo de estructuras con varios elementos se emplean métodos que involucran el uso de matrices de grandes dimensiones, como por ejemplo, el método de elementos finitos, en el cual el crecimiento de las dimensiones de las matrices involucradas en sus cálculos es del orden de $3n$ para problemas bidimensionales y $6n$ para problemas tridimensionales, donde n es el número de nodos que se requieren para definir el modelo matemático de una estructura.

Algunas características importantes que suelen tener estos softwares involucran el uso interno de tablas dinámicas para definir la interacción de los elementos que componen una estructura, tipos de secciones y sus dimensiones, materiales y sus características mecánicas, casos de cargas básicas, casos de combinaciones de cargas, coordenadas para definir la posición de los elementos, y estas características se potencian con herramientas de dibujo asistido por computador para facilitar la gestión de ingreso de información al programa mediante una interfaz gráfica.

Ejemplo de algunos programas conocidos que cuentan con estas características son RISA-2D/3D y SAP2000. A continuación, se listan otros softwares conocidos desarrollados para desempeñarse en el análisis y diseño estructural (The Free Encyclopedia, 2020):

- ABAQUS: software para análisis estructural FEM.
- Advance Design: software BIM para análisis estructural FEM.
- ArchiCAD: software BIM y de modelado 3D aplicado a la ingeniería civil y estructural.
- COMSOL Multiphysics: simulación y física aplicada a la ingeniería estructural.

- Extreme Loading for Structures: software avanzado de análisis estructural no lineal.
- FEATool Multiphysics: simulación y física aplicada a la ingeniería estructural.
- FEMtools: software FEM que provee soluciones de scripting y análisis avanzados para la ingeniería estructural.
- FreeCAD: software suizo open source de propósito genérico con herramientas para ingeniería.
- MicroStation: software BIM y de modelado 3D aplicado a la ingeniería civil y estructural.
- Midas Civil: software FEM para el modelado estructural, análisis y diseño de puentes.
- OpenSees: software de ingeniería sísmica.
- Realsoft 3D: software de análisis y diseño general 3D.
- Revit: software BIM y de modelado 3D aplicado a ingeniería civil y estructural.
- RFEM: software de análisis y diseño general 3D.
- SDC Verifier: software para verificación estructural y chequeo de normativas de acuerdo con diferentes estándares de la industria.
- SimScale: software de simulación física (Dinámica de Fluidos Computacional, FEM y Análisis Térmico) aplicada a la ingeniería civil y estructural.
- SketchUp: software BIM y de modelado 3D aplicado a la ingeniería civil y estructural.
- STAAD: software BIM de análisis y diseño estructural.
- Tekla Structures: software BIM y de modelado 3D aplicado a la ingeniería civil y estructural.
- PLPAK: software BIM de análisis y diseño de losas de hormigón armado y análisis estructural estático y dinámico.

Los programas listados anteriormente comparten el hecho de que siguen en desarrollo para cada vez incorporar más herramientas relacionadas a la ingeniería civil estructural, a la vez que se adaptan a las nuevas tecnologías que permiten hacer cálculos más avanzados en menos tiempo, modernizando desde el trabajo interno de cada uno de los programas hasta la presentación visual que ofrecen, sin embargo, solo uno de ellos es open source y gratuito, siendo este FreeCAD. Otros autores han intentado resolver este inconveniente desarrollando sus propios programas, teniendo como objetivo el compartir sus creaciones con otras personas y para ayudarse en los entornos académicos donde han trabajado, partiendo de la

base común que primero necesitan resolver un modelo matemático para calcular los esfuerzos internos de una estructura.

A continuación, se listan varios softwares que han sido desarrollados buscando tales objetivos, realizar un análisis por el método de elementos finitos (FEM), desarrollándose por medio del Método Matricial de la Rigidez, de los cuales varios han sido abandonados o no han sido actualizados más allá de la versión beta o a tecnologías modernas. Todos tienen la misma iniciativa y propósito, difundir el conocimiento de manera libre y que quienes usen estas herramientas no tengan que lidiar con licencias restrictivas.

- W-Trite (Díaz y Massa, s.f.).
- Frame3DD (Gavin, 2014)
- Delta Beam (Civil Engineering Software Database [CESDb], s.f.).
- bars3d (Petrescu, 2013).
- DS-Frame2D (Petro, 2014).
- PSA (Sourceforge, 2015).
- FRAMEWORK 2D+3D (Wolsink, s.f.).

Otros programas de desarrollo más reciente han sido elaborados, pero siguen manteniendo un estado beta o siguiendo una rutina estática como ejemplo, lo que significa que necesitan mayor desarrollo para alcanzar la autonomía que presentan programas que compiten en el mercado actual.

- Script de análisis modal (Thakkar, 2018).
- anaStruct (Belmont, Smith y Vink, 2021)

Se han encontrado más programas que, si bien se distribuyen como archivos libres, estos son en su mayoría programas dependientes de una plataforma mayor (por ejemplo, una planilla de MS Excel) y, por lo tanto, no cumplen con el objetivo de ser software libre.

6. Cálculo de esfuerzos internos

6.1. Diseño sismo resistente

El análisis sísmico busca establecer exigencias mínimas de resistencia que debe cumplir una estructura para soportar fuerzas producto de las aceleraciones experimentadas por los diferentes elementos que componen a un edificio. El análisis sísmico en Chile se separa en tres categorías: Análisis Estático, Análisis Modal Espectral, Análisis Tiempo Historia. El método descrito en este documento e implementado en β -beam corresponde al Análisis Estático para la distribución del corte basal en la estructura resistente, tal y como es especificado por la norma NCh 433 (1996).

6.1.1.Método estático

El análisis o método estático es una simplificación de un análisis dinámico donde la acción de las aceleraciones sobre las masas de una estructura producto de un sismo trabajan como una carga estática equivalente, actuando sobre el centro de masa de los pisos que compongan la estructura resistente del edificio.

Para llevar a cabo el desarrollo del método estático, se necesita conocer los siguientes parámetros de la estructura en función de su serviciabilidad, zona geográfica y tipo de suelo donde se emplea su construcción (NCh 433, 1996):

- I : Factor de Importancia del edificio según su categoría de uso.
- n, T', S : parámetros relativos al tipo de suelo de fundación.
- A_0 : Aceleración efectiva según la zona sísmica.
- R : Factor de modificación de respuesta.
- T^* : Período con mayor masa traslacional equivalente en la dirección de análisis.
- C : Coeficiente sísmico.

El coeficiente sísmico queda definido por la siguiente ecuación:

$$C = \frac{2.75 \cdot A_0}{g \cdot R} \left(\frac{T'}{T^*} \right)^n$$

El corte basal, Q_0 , se define como:

$$Q_0 = C \cdot I \cdot P$$

Donde P se define como el peso sísmico del edificio en el momento que se espera la ocurrencia del sismo, calculándose como el peso propio de la estructura más un porcentaje de la sobrecarga de uso que se contempla un mínimo entre el 25% y el 50% según la aglomeración de personas que pueda concurrir al edificio en su vida útil.

Para calcular la fuerza aplicada en el centro de masas de cada piso, se utilizan dos fórmulas recursivas donde se define la fuerza acumulada por piso. Se definen A_k y F_k como el porcentaje de participación del corte basal en el nivel k y la fuerza horizontal aplicada en el centro de masas del piso k .

$$A_k = \sqrt{1 - \frac{Z_{k-1}}{H}} - \sqrt{1 - \frac{Z_k}{H}}$$

Donde:

- Z_k : Altura acumulada del nivel “ k ”.
- H : Altura relativa del nivel “ k ”.

$$F_k = \frac{A_k P_k}{\sum_{j=1}^N A_j P_j} Q_0$$

Donde:

- P_k : Peso sísmico del piso “ k ”.

En cada nivel o piso de la estructura, las fuerzas horizontales aplicadas sobre el centro de masa generan una reacción con la misma magnitud y sentido opuesto en el centro de rigidez

del mismo piso. Esto puede generar un efecto amplificador al momento de distribuir la fuerza a cada elemento del piso debido a la existencia de un momento torsor a causa de una excentricidad producida entre el centro de masas y el centro de rigidez. Si se tiene una estructura simétrica, el efecto del momento debido a la excentricidad natural de la estructura es nulo. Adicionalmente, la norma NCh 433 (1996) estipula que debe considerarse una torsión accidental producto del desplazamiento del centro de masa o de la aplicación de momentos de torsión estáticos en cada nivel, para lo cual entrega dos alternativas:

- Desplazamiento transversal del centro de masas: $e_x = \pm b_{kx}$; $e_y = \pm b_{ky}$
- Excentricidad del momento torsor estático: $e_x = \pm 0,1 b_{kx} \cdot \frac{Z_k}{H}$; $e_y = \pm 0,1 b_{ky} \cdot \frac{Z_k}{H}$

La norma permite prescindir de la torsión accidental si se cumple con que el desplazamiento del modelo, con el centro de masas tanto en su punto original como desplazado de este, presenta una variación no mayor al 20% en las deformaciones. Para los fines demostrativos de este documento, y dado que los ejemplos son simétricos, no se aplican los efectos de momentos producto de la excentricidad natural y se desprecian los efectos de momentos producto de la excentricidad accidental.

6.1.2. Período fundamental de la estructura

Se emplea el método de Rayleigh, el que entrega resultados con aproximaciones satisfactorias (Nassani, 2013), para la determinación del período fundamental, T^* :

$$T^* = \frac{2\pi}{\sqrt{g}} \sqrt{\frac{\sum w_i \delta_i^2}{\sum w_i \delta_i}}$$

Donde:

- δ_i y w_i : Desplazamiento horizontal y peso sísmico por piso.
- g : Aceleración de gravedad.

6.2. Análisis matricial de estructuras

El análisis matricial de estructuras contempla métodos para el cálculo de estructuras estáticamente indeterminadas que se benefician de la capacidad de computadores para llevarse a cabo, dado que son procedimientos complejos de desarrollar manualmente. Existen dos métodos ampliamente difundidos: Método Matricial de Rigidez y Método de Flexibilidad. En este documento se utiliza el Método Matricial de Rigidez debido a que su uso es más difundido por ser un método más simple de programar que el Método de Flexibilidad, aunque este último, dado el avance tecnológico y el avance en el área del Cálculo Asistido por Computadores, ha demostrado ser más eficiente y preciso para la misma cantidad de recursos que el Método Matricial de Rigidez. El Método de Flexibilidad se ha reformulado para propósitos de uso con la tecnología moderna bajo el nombre de Método de Fuerza Matricial o “*Matrix Force Method*” (Iran University of Science and Technology, s.f.).

El Método Matricial de Rigidez es un método que viene a reemplazar métodos de análisis clásicos como el Método de Pendiente-Deflexión, desarrollado por George Maney (1914) o el Método de Cross, desarrollado por Hardy Cross (1930), gracias al avance computacional que tuvo lugar cerca de 1960.

6.2.1. Grados de libertad

Para llevar a cabo el Método Matricial de la Rigidez, es necesario hacer una representación idealizada de la estructura en cuestión por medio de elementos que formen una estructura de barras unidas por nodos, permitiendo que el modelo sea resuelto con los principios de la Resistencia de Materiales.

El fundamento teórico viene dado por la representación mediante barras largas elásticas, cuya diferencia de un sólido deformable real son los grados de libertad necesarios para definir sus deformaciones o desplazamientos asociados a infinitos grados de libertad. Para el caso de una barra larga elástica, o prisma mecánico, de longitud grande en comparación a su sección transversal, los desplazamientos son asociables a la curva elástica del material del que se

componen, lo que permite reducir el campo de desplazamientos a un número finito de parámetros, reduciéndose al mismo tiempo los grados de libertad. Para el caso de las barras (representando vigas y columnas), fijándose los extremos queda completamente definido el estado deformado de la misma, permitiendo calcular con una buena aproximación y con un número finito de ecuaciones algebraicas los desplazamientos de un número finito de grados de libertad.

Finalmente, una barra en un sistema coordenado de 2 dimensiones tiene completamente definida su geometría considerando tres grados de libertad por nodo (inicio y fin de la barra), correspondiendo estos a dos desplazamientos y un giro por nodo. Análogamente, para una barra en un sistema coordenado de 3 dimensiones, cada nodo posee tres desplazamientos y tres giros, dando un total de 12 grados de libertad por barra.

6.2.2.Cargas Tributarias

Para la distribución de las cargas que solicitan a una estructura, se analiza por cada elemento que le componga (barra) sus respectivas reacciones de empotramiento perfecto, teniendo un sistema equivalente con una estructura idealizada donde todas las cargas son representadas por fuerzas nodales como se muestra en el siguiente esquema:

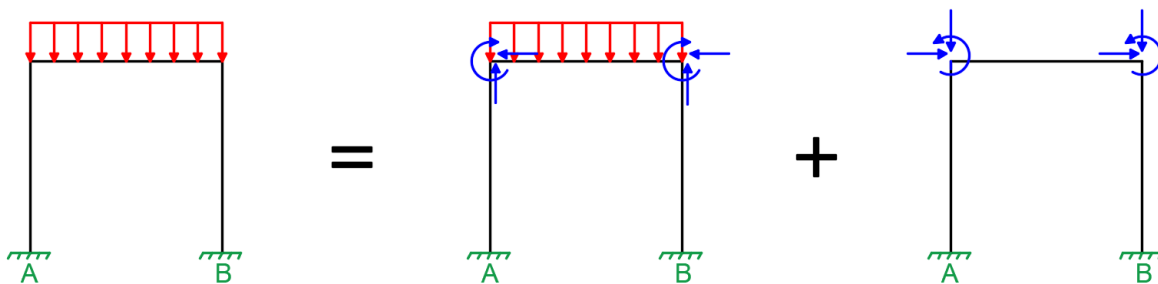


Figura 2: Transformación a un sistema equivalente con cargas puntuales.

Para el caso de las cargas en la superficie de una losa, estas se traspasan a las vigas que la sustentan como cargas trapezoidales, teniendo en cuenta las condiciones de borde de la losa que descansa en el marco. Las fórmulas de empotramiento perfecto que se consideran para el cálculo de reacciones contemplan la superposición de dos cargas con distribución lineal, una en cada extremo de una barra dada, y una carga central con distribución constante.

Reacciones para cargas linealmente distribuidas:

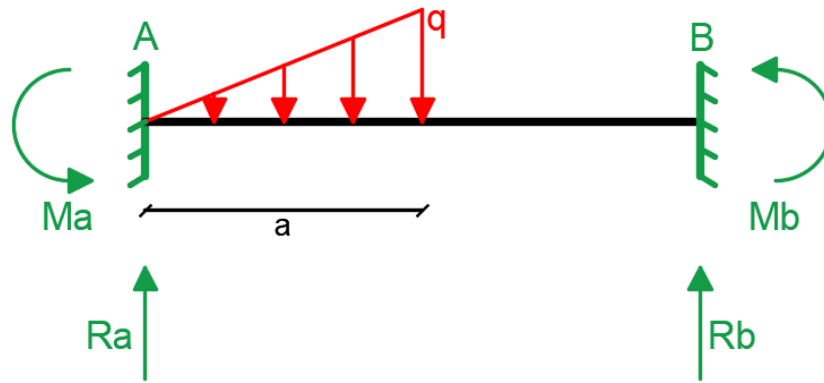


Figura 3: Empotramiento perfecto, carga distribuida constante.

$$R_a = \frac{qa}{20} \cdot \left[10 - \frac{a^2}{L^2} \left(15 - \frac{8a}{L} \right) \right]$$

$$R_b = \frac{qa^3}{20L^2} \cdot \left(15 - \frac{8a}{L} \right)$$

$$M_a = \frac{qa^2}{30} \cdot \left[10 - \frac{a}{L} \left(15 - \frac{6a}{L} \right) \right]$$

$$M_b = -\frac{qa^3}{20L} \cdot \left(5 - \frac{4a}{L} \right)$$

Donde:

- R_a : Reacción vertical en el punto A.
- R_b : Reacción vertical en el punto B.

- M_a : Reacción de momento en el punto A.
- M_b : Reacción de momento en el punto B.
- q : Carga lineal.
- a : Distancia desde el punto A hasta el final de la aplicación de la carga q .
- L : Longitud total de la barra AB.

Reacciones para cargas constantes:

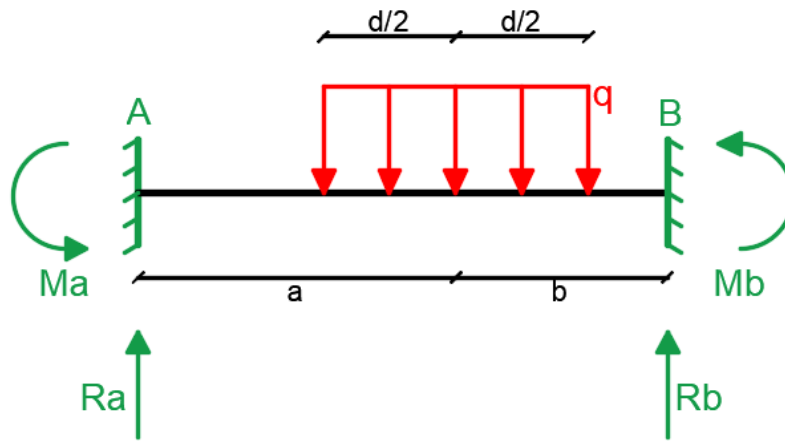


Figura 4: Empotramiento perfecto, carga distribuida constante.

$$R_a = q \cdot \frac{d}{L^3} \cdot \left[(2a + L)b^2 + \frac{a - b}{4} d^2 \right]$$

$$R_b = q \cdot \frac{d}{L^3} \cdot \left[(2b + L)a^2 - \frac{a - b}{4} d^2 \right]$$

$$M_a = q \cdot \frac{d}{L^2} \cdot \left[ab^2 + \frac{d^2}{12} (L - 3b) \right]$$

$$M_b = -q \cdot \frac{d}{L^2} \cdot \left[a^2b + \frac{d^2}{12} (L - 3a) \right]$$

Donde:

- R_a : Reacción vertical en el punto A.

- R_b : Reacción vertical en el punto B.
- M_a : Reacción de momento en el punto A.
- M_b : Reacción de momento en el punto B.
- q : Carga constante.
- a : Distancia desde el punto A hasta la mitad de la aplicación de la carga q .
- b : Distancia desde el punto B hasta la mitad de la aplicación de la carga q .
- L : Longitud total de la barra AB.
- d : Longitud de aplicación de la carga q .

6.2.3. Matriz de rigidez

Cada barra del sistema resistente a analizar aporta rigidez a la estructura. Se define la matriz de rigidez elemental en coordenadas locales, k_p , para una barra p empotrada en ambos extremos:

$$k_p = \begin{bmatrix} \frac{2EI(2 + \beta)}{L(1 + 2\beta)} & \frac{2EI(1 - \beta)}{L(1 + 2\beta)} & 0 \\ \frac{2EI(1 - \beta)}{L(1 + 2\beta)} & \frac{2EI(2 + \beta)}{L(1 + 2\beta)} & 0 \\ 0 & 0 & \frac{AE}{L} \end{bmatrix}$$

Donde:

- E : Módulo de elasticidad longitudinal de la barra.
- I : Inercia en el eje fuerte de la barra.
- L : Largo de la barra.
- A : Área de la sección transversal de la barra.
- $\beta = \frac{6EI\kappa}{GAL^2}$

- κ : Factor de forma de la sección para deformaciones de corte.
- G : Módulo de elasticidad transversal de la barra. $G = \frac{E}{2(1+\nu)}$
- ν : Módulo de *Poisson*.

La matriz de rigidez elemental debe ser transformada a coordenadas globales para poder sumar la participación de cada barra al sistema resistente. Se define la Matriz de Compatibilidad Geométrica de una barra p , a_p , para obtener la Matriz de Rigidez en coordenadas globales de la barra, K_p .

$$a_p = \begin{bmatrix} -s/L & c/L & 1 & s/L & -c/L & 0 \\ -s/L & c/L & 0 & s/L & -c/L & 1 \\ c & s & 0 & -c & -s & 0 \end{bmatrix}$$

$$[K_p] = [a_p]^t [k_p] [a_p]$$

Donde:

- s : Seno del ángulo de inclinación de la barra respecto del eje coordenado de la estructura.
- c : Coseno del ángulo de inclinación de la barra respecto del eje coordenado de la estructura.

Para compilar en una ecuación todas las propiedades de rigideces individuales que represente el comportamiento interno de la estructura, se busca satisfacer la ecuación de la Ley de Hooke de Mecánica de Sólidos de modo matricial, lo que permite generar tantas ecuaciones como sean necesarias para el sistema hiperestático. Estas ecuaciones se obtienen de la ecuación matricial que relaciona las fuerzas con los desplazamientos nodales por medio de la Matriz de Rigidez Total, K_t , que es el resultado de la suma de las submatrices de cada matriz K_p en los índices que relacionan a los nodos A y B de una barra p con los respectivos nodos de la estructura:

$$\{R\} + \{F\} = [K_t]\{\delta\}$$

Donde:

- F : Vector de fuerzas nodales obtenidas de las fuerzas externas aplicadas a la estructura.
- R : Reacciones inicialmente desconocidas.
- K_t : Matriz de rigidez del sistema.
- δ : Vector de desplazamientos nodales.

Finalmente, la ecuación se puede separar en dos subsistemas donde uno solo contempla como incógnita al vector de desplazamientos, permitiendo despejar δ y prescindir de las reacciones en los apoyos que pueden ser calculadas posteriormente:

$$\begin{aligned}\{F\} &= [K_t]\{\delta\} \\ [K_t]^{-1}\{F\} &= \{\delta\}\end{aligned}$$

6.2.4. Tramos rígidos

El método anterior considera una estructura que puede resultar más flexible de lo deseado, por lo que se consideran todos los nodos rígidos. Para evitar obtener una estructura más rígida de lo necesario, se considera un porcentaje de penetración, p , de los tramos flexibles de todos los elementos en los nodos, siendo esta penetración, en cada uno de los extremos de una barra, de un 25% del alto de sus secciones en el nodo para elementos esbeltos de hormigón armado (Guendelman, 2014):

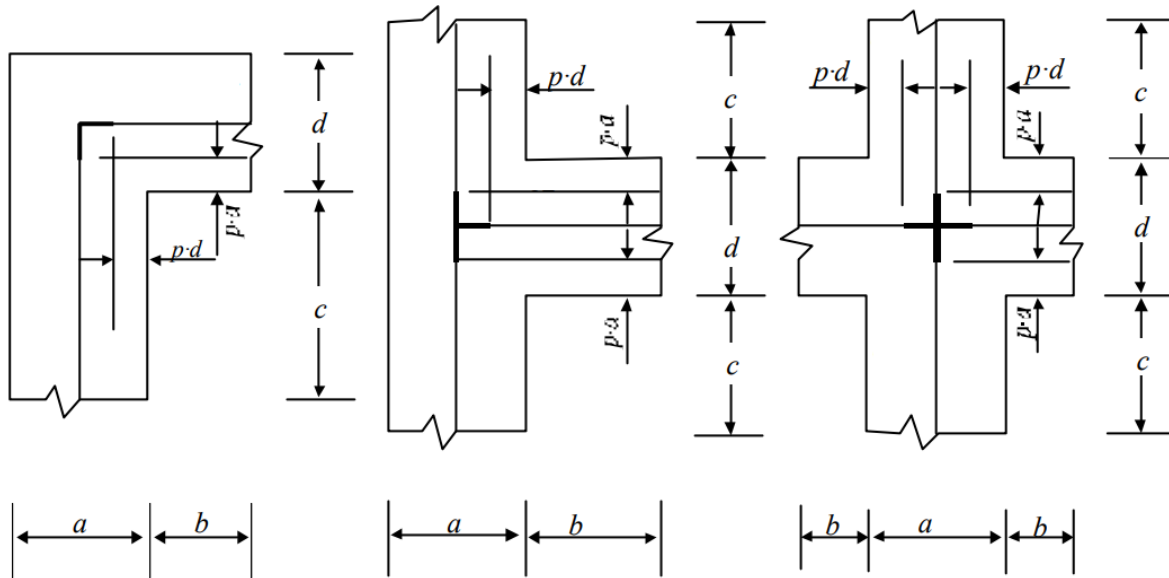


Figura 5: Penetración de elementos en nudos rígidos.

Considerando los tramos rígidos, cambian unas variables previamente definidas para el método matricial de la rigidez como se expone a continuación:

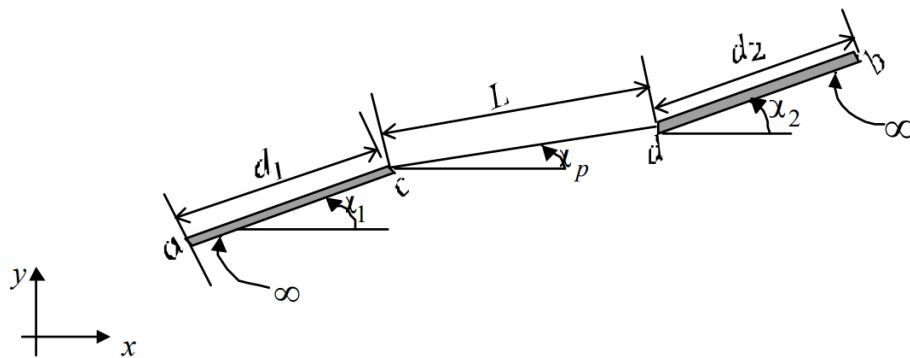


Figura 6: Barra con conexiones infinitamente rígidas.

- k_p se calcula solamente para el tramo flexible de cada barra.
- a_p cambia de la siguiente manera:

$$a_p = \begin{bmatrix} -s_0/L & c_0/L & \left(1 + \frac{d_1}{L}c_1\right) & s_0/L & -c_0/L & \frac{d_2}{L}c_2 \\ -s_0/L & c_0/L & \frac{d_1}{L}c_1 & s_0/L & -c_0/L & \left(1 + \frac{d_2}{L}c_2\right) \\ c_0 & s_0 & d_1s_1 & -c_0 & -s_0 & d_2s_2 \end{bmatrix}$$

Donde:

- $s_0 = \text{sen}(\alpha_p)$
- $s_1 = \text{sen}(\alpha_p - \alpha_1)$
- $s_2 = \text{sen}(\alpha_p - \alpha_2)$
- $c_0 = \text{cos}(\alpha_p)$
- $c_1 = \text{cos}(\alpha_p - \alpha_1)$
- $c_2 = \text{cos}(\alpha_p - \alpha_2)$
- d_1 y d_2 : Extensiones de tramos rígidos.

7. Criterios de diseño

7.1. Consideraciones de los materiales

7.1.1. Consideraciones del hormigón

El módulo de elasticidad del hormigón, para hormigones de peso normal, se obtiene de acuerdo con la ecuación 19.2.2.1.b del código ACI 318S-14 (Comité ACI 318, 2014). Por otro lado, se consideraron las suposiciones del hormigón de acuerdo con el punto 22.2.2.

7.1.2. Consideraciones del acero

De acuerdo con punto 21.2.2.1 del código ACI 318S-14 (Comité ACI 318, 2014), para refuerzo corrugado, ϵ_{ty} debe ser $\frac{f_y}{E_s}$. para refuerzo corrugado Grado 420, se permite tomar ϵ_{ty} igual a 0.002. Siendo esta la deformación utilizada para cualquier caso en el programa.

De acuerdo con Park y Paulay, la curva esfuerzo deformación del acero, puede representarse a través del modelo elastoplástico ideal, que compone una curva de dos tramos, uno elástico y otro tramo infinitamente plástico (1983). (Ver Figura 7).

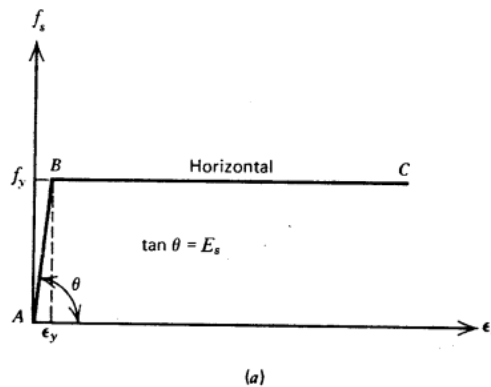


Figura 7: Simplificación de la curva esfuerzo deformación del acero (Park y Paulay, 1983).

7.2. Método de diseño por resistencia

7.2.1.Principios básicos de diseño

Se utiliza el método de diseño LRFD, considerando las disposiciones de algunos puntos del código ACI 318S-14 (Comité ACI 318, 2014). El método por el cual se verifica que un elemento cumpla un esfuerzo determinado es mediante el método de diseño por resistencia, de acuerdo con el punto 4.6.1 del código ACI 318S-14 (Comité ACI 318, 2014) y de acuerdo con el comentario R4.6. El requisito básico para el diseño por resistencia se puede expresar como $\phi S_n \geq U$.

Estos criterios se obtienen de los puntos 9.5.1.1 y 10.5.1.1 del código ACI 318S-14 (Comité ACI 318, 2014) referentes a la resistencia de diseño en vigas y columnas respectivamente.

7.2.2.Combinaciones de carga

Tabla 7.1: Combinaciones de carga del software de acuerdo a la norma NCh 3171 (2010).

Combinación
1.4 D
1.2 D + 1.6 L
1.2 D \pm 1.4 E + L
0.9 D \pm 1.4 E

Se toman como base las combinaciones de carga básicas mencionadas en el punto 9.1.1 de la norma NCh 3171 Of. 2010 (NCh 3171, 2010), modificadas (ver Tabla 7.1), por otro lado, las cargas Lr, R, S y W se consideran nulas.

7.2.3.Factor de reducción de resistencia

De acuerdo con el punto 21.1 del código ACI 318S-14 (Comité ACI 318, 2014), los factores de reducción de resistencia, ϕ , deben cumplir con la Tabla 21.2.1, excepto lo modificado por 21.2.2, 21.2.3 y 21.2.4, siendo aplicables las tablas 21.2.1 y 21.2.2.

7.3. Diseño a esfuerzos axiales, de flexión y flexo-compresión

7.3.1.Generalidades

De acuerdo con el punto 22.4.1 del código ACI 318S-14 (Comité ACI 318, 2014), la resistencia nominal a flexión y carga axial debe calcularse de acuerdo con las suposiciones del punto 22.2.

7.3.2.Resistencia axial a la compresión máxima

P_o , se calcula con la ecuación 22.4.2.2 del código ACI 318S-14 (Comité ACI 318, 2014) y el valor máximo de la carga nominal $P_{n,max}$ está limitado al valor de P_o según la tabla 22.4.2.1, utilizando $0.80P_o$, de acuerdo con el punto a, para los cálculos realizados en este programa.

7.3.3.Resistencia axial a tracción máxima

La resistencia axial a tracción máxima se debe calcular de acuerdo con el punto 22.4.3.1 del código ACI 318S-14 (Comité ACI 318, 2014).

7.4. Diseño al corte

7.4.1.La resistencia al corte de diseño

Se calcula de acuerdo con 9.5.1.1b y 10.5.1.1b del código ACI 318S-14 (Comité ACI 318, 2014), considerando los factores de reducción mencionados anteriormente en el punto 8.1.3.3.

7.4.2.La resistencia nominal

V_n (Resistencia nominal al corte) se calcula de acuerdo con la ecuación 22.5.1.1 del código ACI 318S-14 (Comité ACI 318, 2014), donde el primer componente es V_c (Resistencia al corte que aporta el hormigón), de acuerdo con el punto 22.5.5, y el segundo componente es V_s (Resistencia al corte que aporta el acero), de acuerdo con la ecuación 22.5.10.5.

7.4.3.La cuantía mínima para refuerzo transversal en vigas

Se calcula de acuerdo con la tabla 9.6.3.1 y, en columnas, de acuerdo con la tabla 10.6.2.2 del código ACI 318S-14 (Comité ACI 318, 2014).

7.4.4.Espaciamiento máximo de estribos en vigas

El espaciamiento en la zona de la viga que se encuentra fuera de la rótula plástica, de acuerdo con la tabla 9.7.6.2.2 del ACI 318S-14 (Comité ACI 318, 2014).

7.4.5.Espaciamiento máximo de estribos en columnas

El espaciamiento máximo, para columnas en zonas distintas a la zona de rótula plástica y empalme, se calcula de acuerdo con la tabla 10.7.6.5.2 y el punto 25.7.2.3 del código ACI 318S-14 (Comité ACI 318, 2014).

7.5. Diseño de marcos especiales resistentes a momento

Para el diseño de marcos dúctiles se optó por los marcos más seguros, en este caso los marcos especiales resistentes a momento, y se consideró como referencia el capítulo 18 del ACI 318S-14 (Comité ACI 318, 2014), destacando los puntos 18.6 para el diseño de vigas, 18.7 para el diseño de columnas y 18.8 para el diseño de nodos.

7.5.1.Vigas

- Los límites dimensionales se calculan de acuerdo con el punto 18.6.2.1.
- El refuerzo longitudinal se calcula de acuerdo con el punto 18.6.3, considerando solo los puntos 18.6.3.1, 18.6.3.2 y 18.6.3.3.
- El refuerzo transversal se calcula de acuerdo con el punto 18.6.4 completo.
- La resistencia al corte se calcula de acuerdo con el punto 18.6.5 completo y se toma en cuenta el comentario R18.6.5 para el cortante de diseño para vigas y columnas.

7.5.2.Columnas

- Límites dimensionales se calculan de acuerdo con el punto 18.7.2.
- Resistencia mínima a flexión de columnas se calcula de acuerdo con el punto 18.7.3 y cumpliendo únicamente con el criterio del punto 18.7.3.2.
- Refuerzo longitudinal se calcula de acuerdo con el punto 18.7.4 completo.
- Refuerzo transversal se calcula de acuerdo con el punto 18.7.5 y considerando para ello los puntos 18.7.5.1, 18.7.5.2, 18.7.5.3 y 18.7.5.4, considerando solo los casos *a* y *b* de la tabla 18.7.5.4.

- Resistencia a cortante se calcula de acuerdo con el punto 18.7.6 completo.

7.6. Longitudes de desarrollo y empalmes

7.6.1.Longitud de desarrollo para barras y alambres corrugados a tracción

La longitud de desarrollo para barras y alambres corrugados se calcula de acuerdo con la tabla 25.4.2.2 del código ACI 318S-14 (Comité ACI 318, 2014), cuyos factores de modificación se pueden obtener en la tabla 25.4.2.4.

7.6.2.Longitud de empalmes por traslapo de barras corrugadas a tracción

La longitud de empalmes por traslapo de barras y alambres corrugados a tracción se calcula de acuerdo con la tabla 25.5.2.1 del código ACI 318S-14 (Comité ACI 318, 2014).

7.6.3.Ganchos estándar para el desarrollo de barras corrugadas a tracción

La geometría de los ganchos estándar para el desarrollo de barras corrugadas en tracción se calcula de acuerdo con la tabla 25.3.1 del ACI 318S-14 (Comité ACI 318, 2014) y su diámetro mínimo interior de doblado de acuerdo con la tabla 25.3.2.

Su longitud de desarrollo se calcula de acuerdo con el punto 25.4.3 del código ACI 318S-14 (Comité ACI 318, 2014).

7.6.4.Longitud de desarrollo para barras corrugadas a compresión

La longitud de desarrollo de barras corrugadas y alambres corrugados a compresión se calcula de acuerdo con el punto 25.4.9 del código ACI 318S-14 (Comité ACI 318, 2014).

7.6.5.Longitud de empalmes por traslapo de barras corrugadas a compresión

La longitud de empalme por traslapo en compresión se calcula de acuerdo con el punto 25.5.5 del código ACI 318S-14 (Comité ACI 318, 2014).

8. Resultados

En este capítulo se detallan los resultados obtenidos de las diferentes partes que componen a β -beam, comenzando con una breve representación de cómo funciona, indicada en el ítem 8.1, para luego pasar a detallar las rutinas fundamentales. También se muestran imágenes del programa, presentando todas las etapas en las que tendrá que interactuar el usuario. Más detalles técnicos sobre el programa, como su instalación y ejercicio de prueba, se abordan en los anexos A y B, respectivamente.

8.1. Diagrama de flujo de la aplicación

El diagrama de flujo del programa, el cual se muestra a continuación, presenta una fiel relación con respecto a cómo fue diseñado internamente, no solo permitiendo demostrar su comportamiento básico y resultado esperado, sino que también resalta lo modular del mismo, lo que se tuvo por objetivo al momento de su desarrollo para permitir que las modificaciones futuras sean simples de implementar. De este modo se facilita también para nuevos usuarios el poder familiarizarse con el código y empezar a trabajar con este rápidamente.

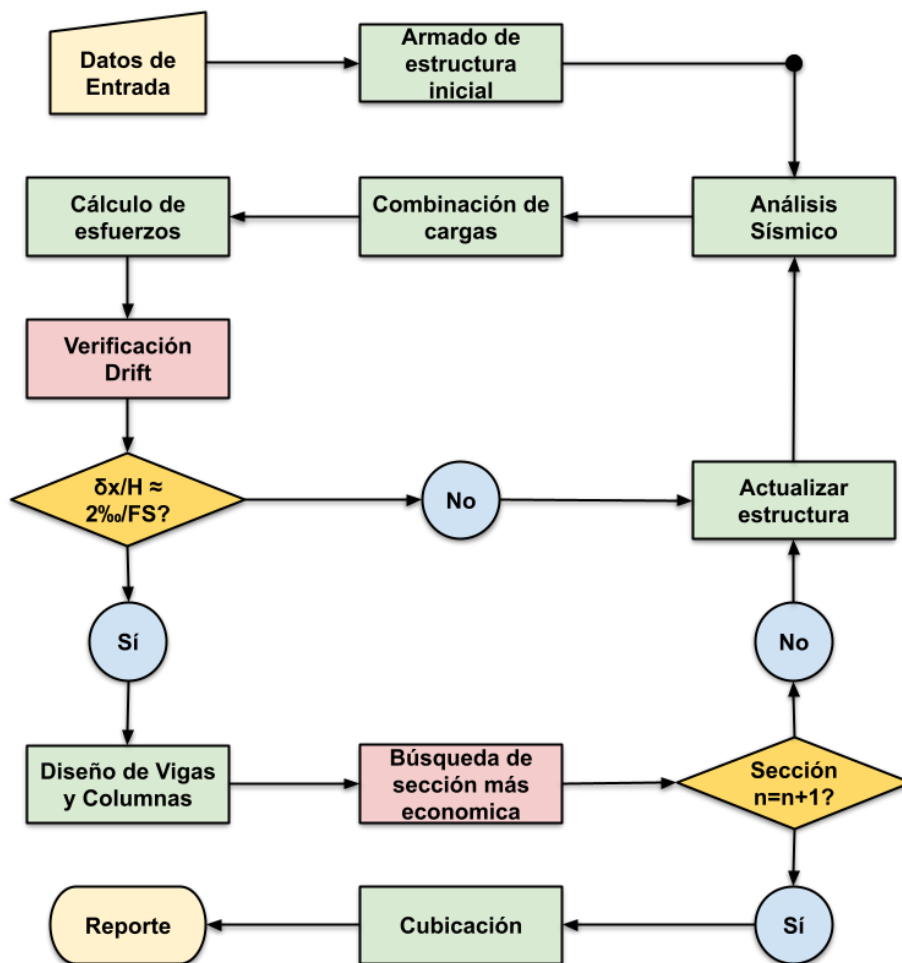


Figura 8: Diagrama de flujo del programa.

8.2. Interfaz y espacio de trabajo

β -beam se maneja en una interfaz de línea de comandos. Se ejecuta desde `cmd.exe` y destina como espacio de trabajo el directorio actual donde se encuentra la consola. Tras ejecutar el programa se analiza el directorio actual en busca de la estructura del espacio de trabajo. En caso de que la carpeta actual no satisfaga las condiciones necesarias, β -beam procede a regenerar el espacio de trabajo con carpetas y archivos con valores por defecto.

El espacio de trabajo contempla los siguientes objetos:

- *src*: Carpeta donde se almacenan archivos generados para la elaboración del reporte.
- *temp*: Carpeta donde se almacenan archivos con información relevante para el desarrollo generado por β -beam, pero no es necesario para el reporte final.
- *input.txt*: Archivo donde se guardan los datos de entrada.
- *reporte.html*: Archivo con el reporte final del análisis.

El espacio de trabajo se presenta a continuación:

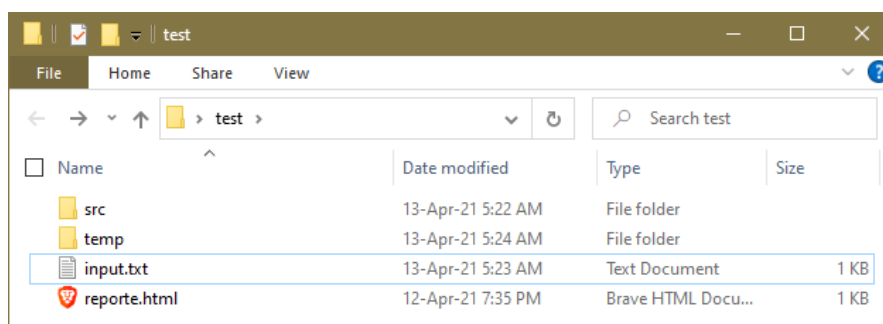
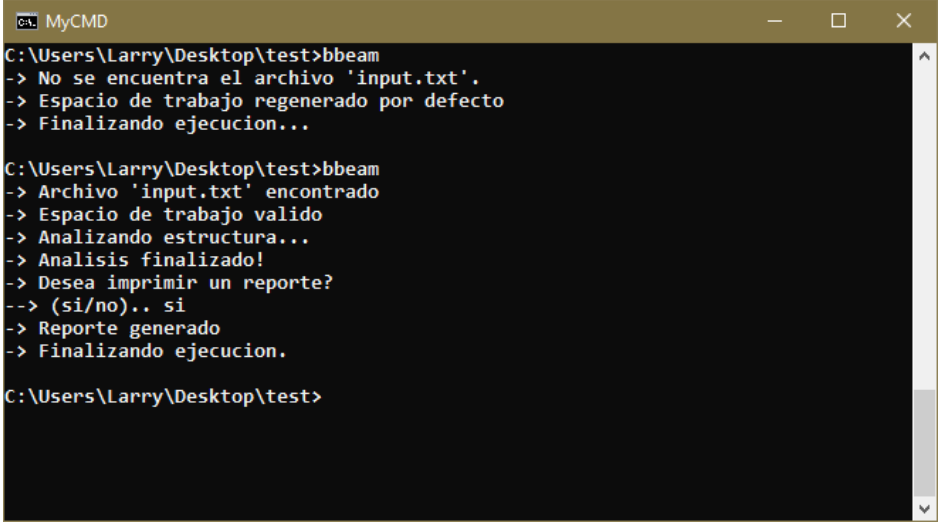


Figura 9: Espacio de trabajo, β -beam.

El espacio de trabajo mínimo contempla todos los archivos y carpetas presentados anteriormente con excepción del archivo *reporte.html*.

Para poder ejecutar el programa, es necesario escribir el comando *bbeam* en la consola. Esto presenta dos únicas instancias, donde una corresponde a la primera ejecución del programa y valida el espacio de trabajo, y la otra corresponde a la ejecución propiamente tal del programa completo, donde se realizan todos los cálculos referentes a la estructura ingresada y se pregunta al usuario si requiere o no imprimir el reporte final en un archivo:



```
MyCMD
C:\Users\Larry\Desktop\test>bbeam
-> No se encuentra el archivo 'input.txt'.
-> Espacio de trabajo regenerado por defecto
-> Finalizando ejecucion...

C:\Users\Larry\Desktop\test>bbeam
-> Archivo 'input.txt' encontrado
-> Espacio de trabajo valido
-> Analizando estructura...
-> Analisis finalizado!
-> Desea imprimir un reporte?
--> (si/no).. si
-> Reporte generado
-> Finalizando ejecucion.

C:\Users\Larry\Desktop\test>
```

Figura 10: Ejecución del programa, β -beam.

8.3. Datos de entrada

Los datos de entrada se manejan en el archivo *input.txt*. En este archivo se introducen todas las variables necesarias para la definición del proyecto. A continuación, se presentan los campos que tienen unidades fijas y deben ser respetadas al momento de ingresar los datos:

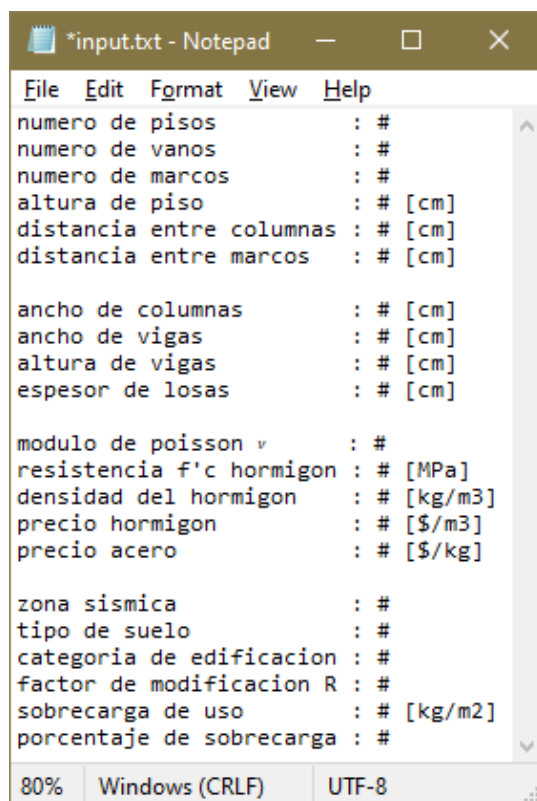


Figura 11: Input de usuario.

El input presenta un formato de formulario, el que tiene 4 secciones destacadas, separadas por saltos de línea doble, los que deben respetarse en conjunto de la cantidad de espacios en blanco y caracteres a la izquierda del carácter “:”, mientras que los valores editables se encuentran a la derecha de estos. Las unidades indicadas anteriormente son referenciales, no pueden ser cambiadas y deben mantenerse separadas un espacio como se indica en la imagen para que β -beam no detecte errores en el archivo. Los numerales representan la posición de los caracteres alfanuméricos que deben ser ingresados en cada campo. Los campos por llenar son los siguientes:

1. Geometría de la estructura:

- **numero de pisos:** Cantidad de niveles que tendrá la estructura.
- **numero de vanos:** Cantidad de vanos que tendrá cada marco de la estructura.
- **numero de marcos:** Cantidad de marcos que tendrá la estructura.

- **altura de piso:** Distancia entre niveles de nodo a nodo (cm).
 - **distancia entre columnas:** Distancia entre columnas de eje a eje (cm).
 - **distancia entre marcos:** Distancia entre marcos de eje a eje (cm).
2. Geometría de los elementos:
- **ancho de columnas:** Dimensión inicial de la sección de todas las columnas (cm).
 - **ancho de vigas:** Ancho inicial de la sección de todas las vigas (cm).
 - **altura de vigas:** Altura inicial de la sección de todas las vigas (cm).
 - **espesor de losas:** Espesor de losas para todos los niveles (cm).
3. Propiedades y precios de los materiales:
- **modulo de poisson ν :** 0.25 para elementos de hormigón.
 - **resistencia f'_c hormigon:** Resistencia a la compresión del hormigón medida en mega pascales (MPa).
 - **densidad del hormigon:** Densidad del hormigón expresada en kilogramos de fuerza por metro cubico.
 - **precio hormigon:** Precio por unidad de volumen (m^3) del hormigon.
 - **precio acero:** Precio por unidad de masa (kg) del acero.
4. Parámetros sísmicos y sobrecarga de uso según NCh 433 (1996) y NCh 1537 (2009):
- **zona sismica:** Zona sísmica (1, 2 o 3).
 - **tipo de suelo:** Categoría del suelo (A, B, C, D o E).
 - **categoria de edificacion:** Categoría de ocupación del edificio (I, II, III o IV).
 - **factor de modificacion R:** Factor de reducción de respuesta.
 - **sobrecarga de uso:** Sobrecarga de uso para las losas expresada en kilogramos de fuerza por metro cuadrado (kg/m^2).
 - **porcentaje de sobrecarga:** Porcentaje de la sobrecarga de uso para la obtención de la masa sísmica del edificio expresado como factor (entre 0 y 1).

Internamente el programa convierte todas las unidades a kilogramos de fuerza y centímetros, pero la presentación de estos datos y otros resultados varía para mejorar la calidad visual de los datos de salida como los que se exponen en el reporte final.

En el caso de que el programa detecte errores en el formato del archivo, este será sobrescrito, borrando toda la información que se encuentre actualmente dentro y volviendo a generar los valores por defecto.

8.4. Armado del marco

Una vez el usuario corre el programa con datos de entrada válidos, el programa genera todos los elementos para definir un modelo matemático de un marco regular plano con elementos de secciones rectangulares.

El programa genera esquemas de referencia para la estructura indicada y especifica el marco del cual se está haciendo el análisis como se ve en las siguientes figuras:

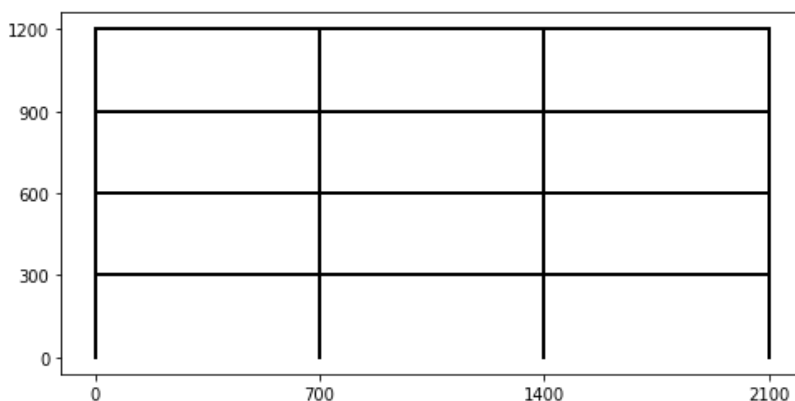


Figura 12: Esquema de marco en 2D.

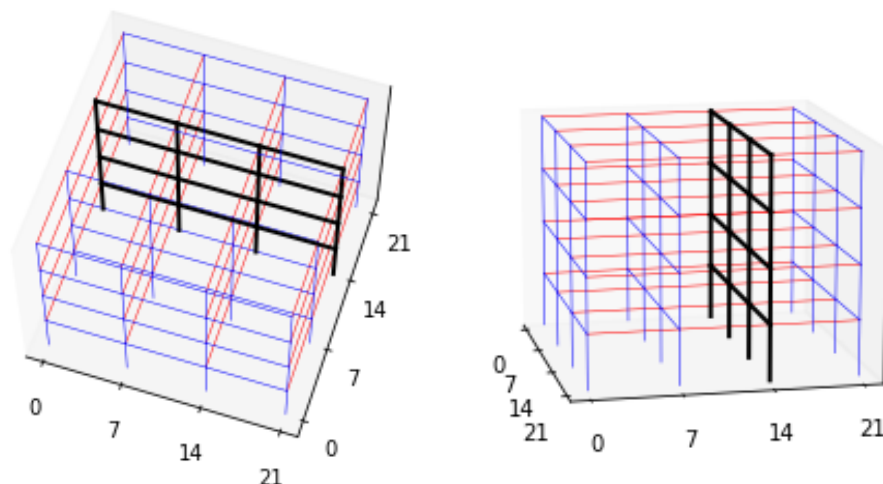


Figura 13: Esquema de marco en 3D.

8.5. Casos de cargas básicas

Una vez se ha generado el marco, se procede a calcular el peso propio de la estructura en función de las dimensiones establecidas para vigas, columnas y espesor de losa (este último permaneciendo constante a lo largo de toda la ejecución del programa). Este valor se calcula por piso.

Para calcular el peso propio de una losa y la sobrecarga que tributa sobre las vigas, se ha tomado en cuenta las condiciones de borde de la losa, verificando si existe o no continuidad, para establecer los ángulos de trapecios que representen las áreas tributarias de las fuerzas distribuidas transmitidas de la losa a las vigas. Para esto, se establece el criterio de que, si existe continuidad de la losa, se considera el lado empotrado, asignándole un ángulo adyacente de 60° si el lado siguiente no posee continuidad. Si dos lados contiguos presentan la misma condición de borde, entonces comparten un ángulo de 45° como se ilustra a continuación:

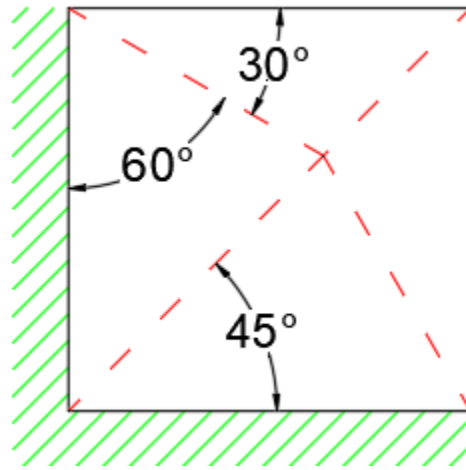


Figura 14: Ángulos para áreas tributarias.

Esto se ha implementado para 4 casos diferentes de losas, lo que satisface el poder calcular el marco intermedio de una estructura mínima compuesta por 2 vanos y 3 marcos. Los casos se muestran a continuación, para un ejemplo de una estructura genérica con 4 marcos y 3 vanos con vista en planta:

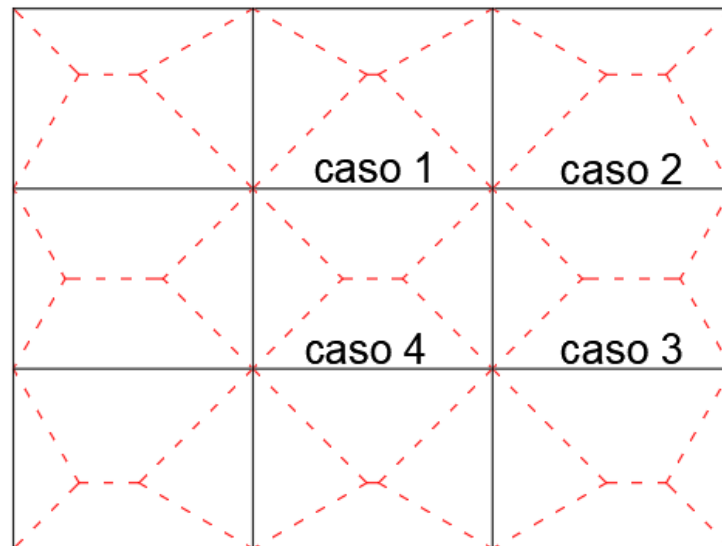


Figura 15: Casos de áreas tributarias.

Por medio de las cargas trapezoidales se puede definir cualquier patrón de carga, pudiendo ser rectangular o triangular, parametrizando el área de las losas como se muestra en la siguiente figura:

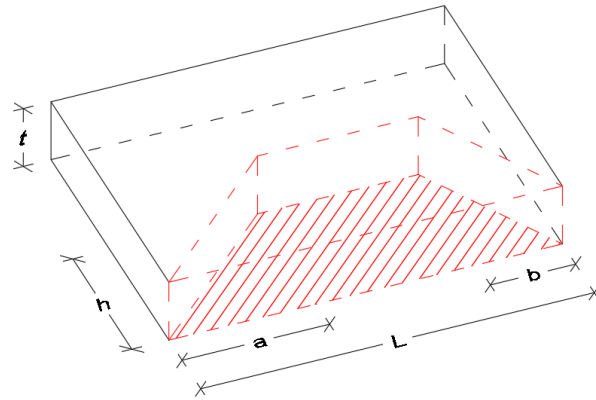


Figura 16: Parámetros área trapezoidal.

Donde:

- h : Altura del trapecio correspondiente a la fuerza distribuida en área.
- L : Largo de la viga.
- a : Largo desde el inicio de la carga lineal hasta la carga constante.
- b : Largo desde el fin de la carga constante hasta el fin de la carga lineal.

8.6. Análisis sísmico

La implementación del análisis sísmico de β -beam está completamente basada en la norma NCh 433 (1996), utilizando el método estático para la repartición de las fuerzas horizontales. A continuación, se resumen los ítems de la norma que han sido integrados al programa:

- 5.5.1: Cálculo de la masa sísmica.
- 5.9.2: Desplazamiento máximo entre dos pisos consecutivos (2‰).
- 6.2.3: Ecuación corte basal.

- 6.2.3.1: Ecuación coeficiente sísmico.
- 6.2.3.1.1: Ecuación coeficiente sísmico mínimo.
- 6.2.3.1.2: Ecuación coeficiente sísmico máximo.
- 6.2.5: Ecuaciones para la distribución del corte basal en altura. Las restricciones las debe tener consideradas previamente el usuario.
- Tabla 6.1: Valor del coeficiente I .
- Tabla 6.2: Valor de la aceleración efectiva A_0 .
- Tabla 6.3: Valor de los parámetros que dependen del tipo de suelo.
- Tabla 6.4: Valores máximos del coeficiente sísmico C .

El peso sísmico de la estructura se calcula en base al peso propio *completo* de esta y al porcentaje especificado por el usuario en los datos de entrada para considerar los efectos de la sobrecarga en el peso total. El peso total se distribuye equitativamente sobre los marcos.

Teniendo en cuenta lo anterior, aun se requiere del criterio del usuario para identificar si la estructura cumple con los requisitos para aplicar el método estático, pues estos no han sido integrados a la aplicación a la fecha de publicación de este documento. Por esto, se recomienda al lector tener presente el ítem 6.2.1 de la norma NCh 433 (1996), para verificar que se cumplen los requisitos mínimos con la estructura. La verificación de los desplazamientos relativos por nivel se trata en el ítem 8.7 del presente informe.

8.7. Verificación de drift

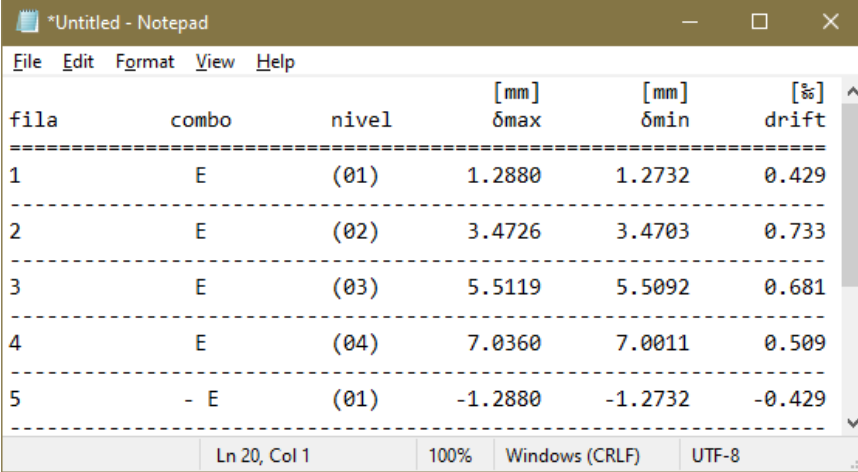
Para comenzar a optimizar la estructura, se verifican los desplazamientos máximos relativos entre pisos, como se indica en el ítem 5.9.2 de la norma NCh 433 (1996), con el objetivo de aproximar el drift máximo al límite establecido. Para los cálculos se aplica un factor de seguridad para reducir la tolerancia que establece la norma de un 2%:

$$\frac{\delta_{n+1} - \delta_n}{H} \leq \frac{0.002}{FS}$$

Donde:

- δ_n : Desplazamiento del nivel n .
- H : Altura entre pisos.
- FS : Factor de seguridad por omisión de los efectos de torsión, equivalente a 1.2.

En cada iteración de la aplicación, se calculan los drifts, lo que resulta en información como la que se muestra a continuación, siendo esta la primera condicionante de optimización:



fila	combo	nivel	[mm] δ_{max}	[mm] δ_{min}	[%] drift
1	E	(01)	1.2880	1.2732	0.429
2	E	(02)	3.4726	3.4703	0.733
3	E	(03)	5.5119	5.5092	0.681
4	E	(04)	7.0360	7.0011	0.509
5	- E	(01)	-1.2880	-1.2732	-0.429

Figura 17: Drifts por nivel.

Posteriormente, se actualizan las secciones de la estructura y todos sus atributos hasta lograr el drift máximo, multiplicando las áreas con el siguiente factor:

$$\psi = \sqrt[4]{\frac{d_k}{d_{max}}}$$

Donde:

- d_k : drift del nivel k .
- $d_{max} = \frac{drift_{max}}{FS}$

8.8. Diseño estructural de marcos especiales resistentes a momento

La segunda parte del programa es el diseño estructural del marco analizado. En esta etapa se reciben los esfuerzos internos calculados en la fase anterior, una vez se cumple con el drift máximo, y se diseñan los elementos capaces de resistir estas solicitaciones con un enfoque en la obtención del menor costo, tomando como base los criterios de diseño del capítulo 7 de esta memoria más otros que son explicados más adelante.

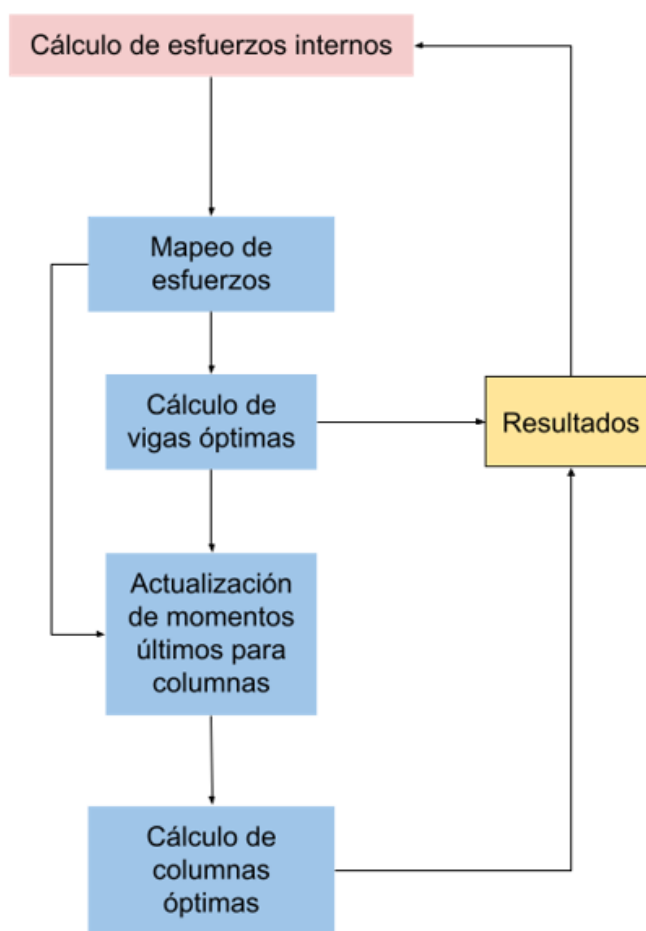


Figura 18: Diagrama de flujo más detallado desde los resultados del cálculo estructural.

En esta fase se recibe la información proveniente del cálculo de esfuerzos internos de la fase anterior, la que incluye los esfuerzos axiales, de corte y momento en cada nodo para cada una de las combinaciones evaluadas, así como las dimensiones de cada elemento y los datos generales de los materiales, como sus resistencias y costos. De esta forma, se asignan los esfuerzos según su tipo a cada elemento en una matriz (mapeo de datos), donde son posicionados de acuerdo con su piso y número de elemento, siendo para las vigas un tipo de elemento por piso y dos tipos para las columnas, siendo estas últimas una para las columnas en los extremos y otra para las columnas al centro. Se busca como entrada la condición más desfavorable de los elementos de un mismo tipo y se calculan todos por igual. El esquema de esta fase se puede ver en la Figura 18.

Con la información anterior ya ordenada, se calculan las vigas óptimas y se genera otra fuente de datos de entrada para evaluar los esfuerzos de momento en los nodos que ingresan a las columnas y compararlos con los momentos nominales de las vigas para así cumplir con el criterio 18.7.3.2 del ACI 318S-14 (Comité ACI 318, 2014), ajustando linealmente su valor en cada extremo del nodo. Luego, se calculan las columnas óptimas y se envían las dimensiones de las secciones calculadas de vigas y columnas a la fase anterior, volviendo a calcular los esfuerzos internos y hasta volver a verificar los drifts, para ajustar los valores en un proceso iterativo hasta tener una respuesta favorable. Finalmente, se genera un reporte donde se muestran los resultados de los esfuerzos internos, sus gráficos, algunos esquemas, los drifts, detallamiento proveniente del análisis estructural y la cubicación.

En este punto se explica brevemente las funciones más relevantes, tomando en consideración las funciones de costo involucradas y la verificación de los esfuerzos, estas etapas son *Mapeo de datos*, *Cálculo de vigas óptimas*, *Corrección de momentos en columnas* y *Cálculo de columnas óptimas*, tratadas en los ítems 8.8.1 al 8.8.4 inclusive.

8.8.1. Mapeo de datos

Se reciben los datos con datos generales como la resistencia a la compresión del hormigón (f_c'), costo del hormigón y acero, número de pisos, número de vanos y datos propios de cada elemento como su largo, dimensiones de la sección previas con los que fueron calculados los esfuerzos y los resultados en los nodos de dichos esfuerzos, donde son estos últimos los que se filtran para poder analizar cada elemento, empezando por las columnas.

Los resultados de los esfuerzos se pueden visualizar en la Tabla 8.1. Cabe mencionar que el formato es el mismo tanto para vigas como para columnas.

Tabla 8.1: Extracto de tabla de resultados de esfuerzos internos de los elementos.

fila	perfil	combo	nodo	[kg] axial	[kg] corte	[kg-m] momento
1	COL 1	1.4 D	(i)	83310.57	-3518.34	-3901.63
2			(j)	74805.57	-3518.34	6653.39
3	COL 1	1.2 D + 1.6 L	(i)	119770.75	-6456.68	-7152.09
4			(j)	112480.75	-6456.68	12217.94
5	COL 1	1.2 D + L	(i)	101635.11	-5166.32	-5724.15
6			(j)	94345.11	-5166.32	9774.80
7	COL 1	1.2 D + 1.4 E + L	(i)	60104.67	23702.1	75561.24
8			(j)	52814.67	23702.1	4454.95
9	COL 1	1.2 D - 1.4 E + L	(i)	143165.56	-34034.73	-87009.54
10			(j)	135875.56	-34034.73	15094.66
11	COL 1	0.9 D + 1.4 E	(i)	12026.35	26606.62	78777.19
12			(j)	6558.85	26606.62	-1042.68
13	COL 1	0.9 D - 1.4 E	(i)	95087.24	-31130.2	-83793.58
14			(j)	89619.74	-31130.2	9597.03
15	COL 2	1.4 D	(i)	127153.76	231.63	134.78
16			(j)	118648.76	231.63	-560.10
17	COL 2	1.2 D + 1.6 L	(i)	197014.79	635.78	469.78

En los puntos 8.3.1.1 y 8.3.1.2 se revisan los datos de entrada seleccionados para vigas y columnas respectivamente.

8.8.1.1. Datos de vigas

Los datos de todas las vigas se almacenan en una lista de listas que simula ser una matriz de elementos ordenados por pisos y número de elemento correspondiente a un determinado piso, tal y como en la matriz que se presenta a continuación:

$$VIG = \begin{bmatrix} Vig_{1,1} & \cdots & Vig_{1,nv} \\ \vdots & \ddots & \vdots \\ Vig_{np,1} & \cdots & Vig_{np,nv} \end{bmatrix}$$

Esta matriz de dimensiones $N \times M$ considera N pisos y M vanos, por lo que hay un elemento por vano en cada piso y estos elementos, a su vez, contienen los datos de entrada para cada viga ($Vig_{n,m}$) que se divide en los datos de cada uno de sus nodos y se puede representar de la siguiente manera:

$$Vig_{n,m} = [Vig_{i,n,m} \quad Vig_{j,n,m}]$$

Estos elementos, a su vez, contienen otros elementos y, tomando del nodo i , por ejemplo, se visualizan como:

$$Vig_{i,n,m} = [V_{u_max} \quad V_{ue_max} \quad M_{u+} \quad M_{u-} \quad V_{u,1.2D+L} \quad l_o \quad VU \quad VUE]$$

Donde V_{u_max} es el corte máximo proporcionado por combinaciones de carga estáticas, V_{ue_max} es el corte máximo proporcionado por combinaciones de carga sísmicas, M_{u+} corresponde al mayor momento positivo, M_{u-} al mayor momento negativo, $V_{u,1.2D+L}$, el corte generado por la combinación de cargas $1.2 D + L$, para ser analizado en el diseño al corte por capacidad, l_o es el largo libre de la viga, y VU y VUE son listas de los cortes con combinaciones estáticas y sísmicas, esto es:

$$VU = [V_{u,1.4D} \quad V_{u,1.2D+1.6L}]$$

$$VUE = [V_{u,1.2D+1.4E+L} \quad V_{u,1.2D-1.4E+L} \quad V_{u,0.9D+1.4E} \quad V_{u,0.9D-1.4E}]$$

Con estos datos, y otras variables globales como los datos generales mencionados anteriormente, se puede diseñar cada viga y encontrar su combinación de materiales óptima.

8.8.1.2. Datos de columnas

Los datos de todas las columnas se almacenan en una lista de listas que simula ser una matriz de elementos ordenados por pisos y número de elemento correspondiente a un determinado piso, tal y como en la matriz que se presenta a continuación:

$$COL = \begin{bmatrix} Col_{np,1} & \cdots & Col_{np,nv+1} \\ \vdots & \ddots & \vdots \\ Col_{1,1} & \cdots & Col_{1,nv+1} \end{bmatrix}$$

Esta matriz de dimensiones $N \times (M + 1)$ considera N pisos y M vanos más uno, por lo que el número de elementos en cada piso corresponda al total de vanos más uno. Estos elementos, a su vez, contienen los datos de entrada para cada viga ($Col_{n,m}$), tomando el máximo de cada una de sus combinaciones de carga y ambos nodos para obtener una única lista:

$$Col_{n,m} = [P_{u_max} \quad P_{u_min} \quad V_{u_max} \quad V_{ue_max} \quad M_{u_max} \quad M_{u_min} \quad H_o]$$

Donde P_{u_max} y P_{u_min} son los esfuerzos axiales máximos y mínimos respectivamente, V_{u_max} y V_{ue_max} son los cortes máximos en valor absoluto para las combinaciones estáticas y sísmicas, M_{u_max} es el momento último máximo, M_{u_min} es el momento último mínimo correspondiente a la combinación de carga de P_{u_min} , y H_o es la altura libre de la columna.

8.8.2. Cálculo de vigas óptimas

8.8.2.1. Cálculo de refuerzo longitudinal

En las vigas se verifica flexión simple, que involucra únicamente flexión y cortante, despreciando los esfuerzos axiales. Por otro lado, se espera que las vigas tengan una falla dúctil, por lo que se impone en el programa que la deformación de las barras de acero en tracción ε_t , sea de al menos 0.005.

Para calcular el área de acero de una sección rectangular se calcula la cuantía mecánica ω de la ecuación (C.10) y se despeja el área A_s proveniente de la ecuación (C.4) quedando como:

$$A_s = \omega \cdot \frac{0.85f'_c b_w d}{f_y}$$

Para los momentos últimos positivo y negativo, y así verificar las cuantías de acuerdo con el punto 9.3.3 del código ACI 318S-14 (Comité ACI 318, 2014), considerando el área de acero correspondiente a cada momento como en la Figura 19b y Figura 19c.

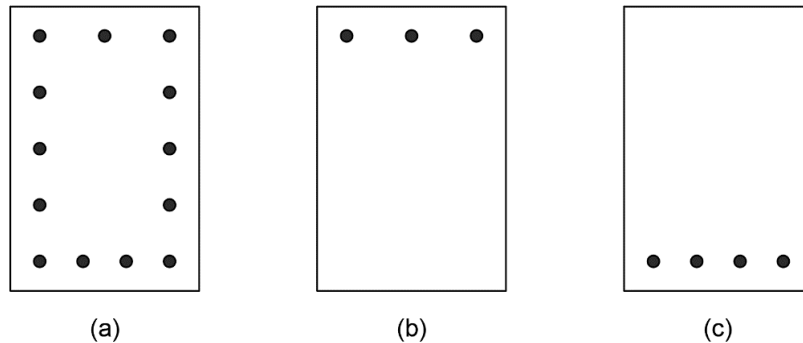


Figura 19: Zonas donde se debe cumplir con las cuantías mínimas.

Estas áreas se dividen en A1 para los momentos positivos y A2 para momentos negativos, lo por que A1 se divide en dos niveles, teniendo al menos la mitad del área en la parte superior para que el resto del área pueda ser complementado por el siguiente nivel en las zonas que se

requiere, separadas a al menos 2.5 veces el diámetro de su diámetro mayor. Lo anterior considerando barras laterales que faciliten la construcción de estos elementos, procurando que no existan separaciones mayores a 25 centímetros ni menores a 20 centímetros entre estas últimas. Por otro lado, las barras principales no estarán separadas horizontalmente a más de 10 centímetros ni a menos de 15 centímetros. En la Figura 20 se puede ver un esquema general del corte transversal en una viga,

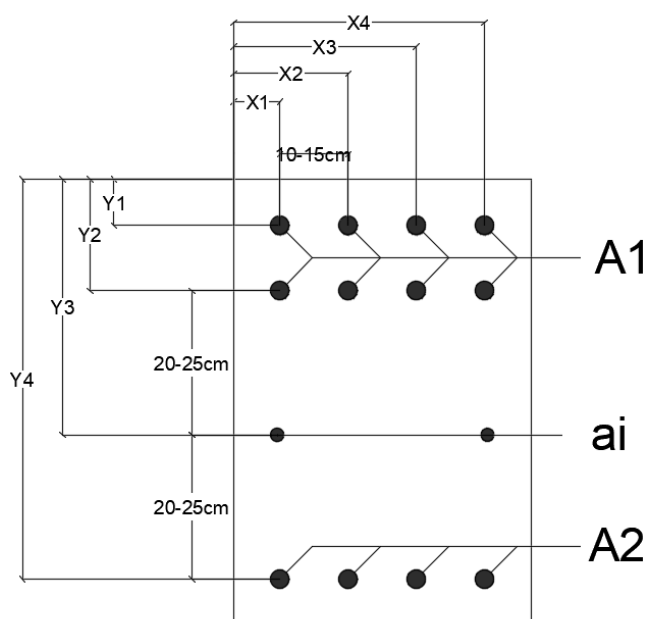


Figura 20: Corte transversal de una viga.

Una vez que se tienen las áreas principales, se calculan todas las áreas de las combinaciones de barras que cumplan con los parámetros anteriores para cada nivel, teniendo en consideración solo dos tipos de barras por nivel y que entre ellas no exista una diferencia de más de 5 mm de diámetro, tomando así el área con menos diferencia respecto a la requerida, luego se verifica nuevamente su resistencia, considerando la sección en ambos sentidos de acuerdo al signo del momento último a evaluar y se calcula su factor de utilización, tomando el mayor valor de las dos verificaciones y limitándolo a un rango entre 75 y 99.9, el cociente entre el momento último requerido y el nominal multiplicado por cien. Todas las combinaciones de ancho y alto de vigas con los diámetros ya minimizados sirven como datos

de entrada para calcular el corte de las vigas y posteriormente se evalúa su costo para conservar la sección más económica que cumpla estas solicitaciones.

8.8.2.2. Cálculo de refuerzo transversal

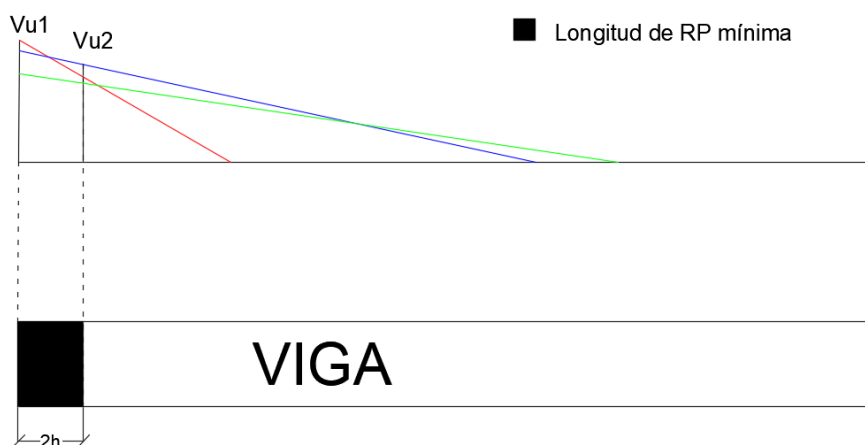


Figura 21: Envolvente del corte ampliando zona de rótula plástica.

Para determinar el corte de diseño, se evalúa el valor absoluto máximo de cada una de las combinaciones, los que son datos de entrada, considerando ambos nodos de manera que se diseñe una viga simétrica y, en base a esto, se define una longitud de rótula plástica (RP), cuya longitud mínima es 2 veces la altura de la viga. En la Figura 21 se muestra la envolvente de posibles cortes en valor absoluto y la longitud de la rótula plástica de la viga.

Los cortes últimos deben ser divididos por el factor de minoración según corresponda su combinación de carga y, para el caso del corte por capacidad, se debe considerar un factor $\phi = 0.75$ y $V_c = 0$ en la zona de rótula plástica, los momentos probables y el corte producido por la combinación de carga estática $1.2 D + L$.

Para calcular el número de ramas por estribo se calcula el mínimo de ramas necesario para que no exista una distancia mayor a 30 centímetros entre las barras horizontales, pese a que el límite sea de 35 cm, como menciona la ecuación 18.7.5.3 del código ACI 318S-14 (Comité

ACI 318, 2014), donde el valor de h_x no debe exceder los 35 cm, hasta el máximo posible, que es una rama por barra. Estos estribos se disponen de forma simétrica, en otras palabras, de la mitad de la sección hacia la izquierda se tiene la misma cantidad de barras que de la mitad hacia la derecha y a la misma distancia.

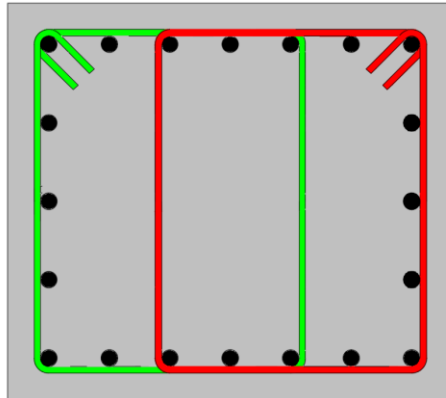


Figura 22: Ejemplo de estribos entrelazados.

Esto tiene un límite que, si bien no aplica al código vigente, se considera una buena práctica. Es el límite de corte de acuerdo con el punto 11.5.7.9 del código ACI 318-05 (ACI Committee, 2005), con una tolerancia superior del 10% para no restringir tanto las posibilidades.

Por otro lado, de acuerdo con el número de ramas, se tiene una cantidad de estribos considerando no más de 2 entrelazados y una traba central en caso de tener un número de ramas impar. Se muestra un ejemplo de estribos entrelazados en la Figura 22.

Por último, se considera el largo de cada uno de los estribos y trabas, recordando las exigencias para ganchos estándar para cada curva.

Con todo esto, se puede construir una función de costo lo suficientemente completa para evaluar una combinación de materiales óptima para la verificación de esfuerzos al corte en vigas. Esto viene siendo:

Costo zona 1

$$= 2 \cdot N^{\circ} \text{ estibos1} \cdot \sum \text{Largos 1} \cdot \text{Área barra 1} \cdot \$\text{acero}$$

$$\text{Costo zona 2} = (N^{\circ} \text{ estibos2} + N^{\circ} \text{ estribos3}) \cdot \sum \text{Largos 2} \cdot \text{Área barra 2} \cdot \$\text{acero}$$

$$\text{Costo estribos} = \text{Costo zona 1} + \text{Costo zona 2}$$

Donde:

En zona de rótula plástica (RP):

Costo zona 1: representa el costo en las zonas de RP

$2 \cdot N^{\circ} \text{ estibos}$: el número de estribos para ambos extremos de la zona de RP

$\sum \text{Largos 1}$: la suma de los largos de estribos y trabas de esta zona

Área barra 1: el área de la barra utilizada para construir los estribos y trabas.

El resto de la viga:

Costo zona 2: representa el costo en la zona no cubierta por la zona de RP

$N^{\circ} \text{ estibos2}$: el número de estribos de esta zona sin incluir empalme

$N^{\circ} \text{ estibos3}$: el número de estribos de la zona de empalme

$\sum \text{Largos 2}$: la suma de los largos de estribos y trabas de esta zona

Área barra 2: el área de la barra utilizada para construir los estribos y trabas.

\$**acero** el costo en pesos chilenos del acero por centímetro cúbico, obteniendo así el costo parcial de este elemento.

Costo estribos: La suma del *Costo zona 1* y *Costo zona 2*

Finalmente, de las combinaciones que verifiquen el esfuerzo de corte, se guarda la información del último valor mínimo, comenzando con un valor mínimo insuperable, como lo es \$9999999 y, en caso de no tener ninguna combinación que verifique, la función devuelve un cero.

8.8.2.3. Función de costo total para vigas

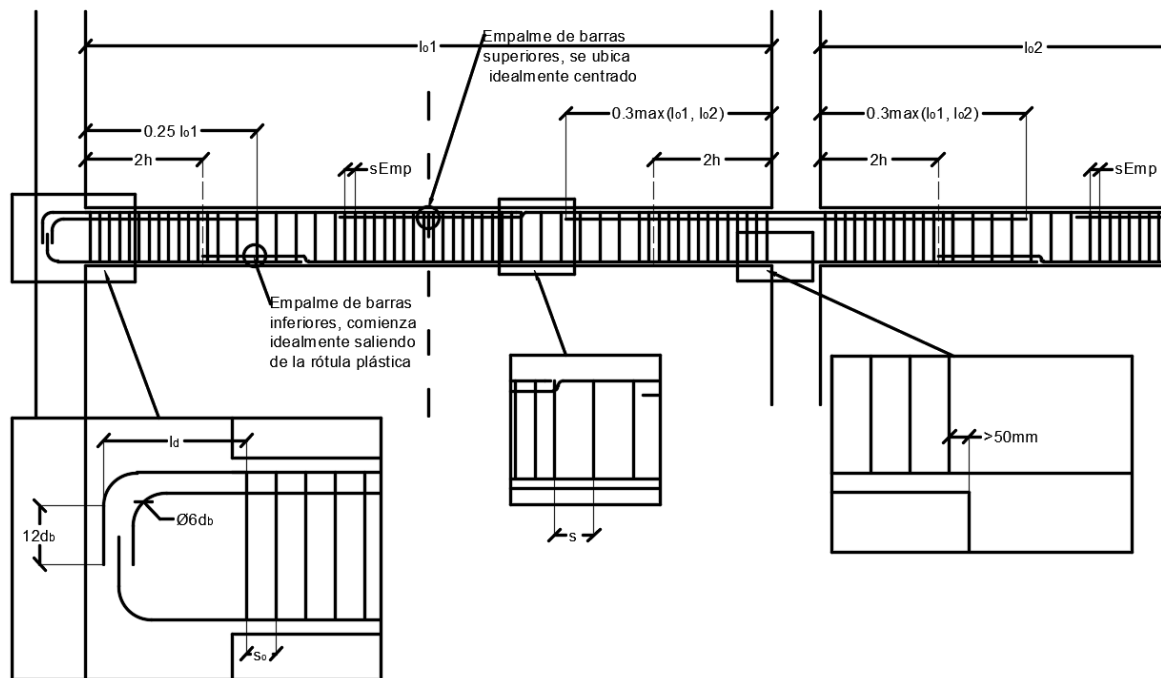


Figura 23: Detalle de viga en marco plano.

La función de verificación de armadura transversal está contenida en la función de verificación de armadura longitudinal y, previo a encontrar un valor mínimo, se requiere verificar ambos esfuerzos, por lo que si una combinación de materiales para verificación al corte no tiene una solución, esta retorna un cero y pasará a la siguiente y, si no existe una combinación que verifique ninguno de los esfuerzos, la función de optimización de vigas es la que devuelve un cero, para activar una señal donde se advierte al usuario que no se pueden verificar las condiciones propuestas. En caso de que existan combinaciones que verifiquen, se aplica el mismo método que en el punto anterior para guardar los datos del valor mínimo y, para este caso, la función de costo total es:

$$\begin{aligned} \text{Costo total} = & \text{Costo estribos} \cdot N^{\circ} \text{vigas} + \text{Costo suples} + \text{Costo laterales} \\ & + \text{Costo armadura superior} + \text{Costo armadura inferior} + N^{\circ} \text{Vigas} \\ & \cdot b_w \cdot h \cdot l_o \cdot \$\text{hormigón} \end{aligned}$$

Y desglosando cada uno de estos costos:

Suples:

Su largo, se considera el mayor entre su longitud de desarrollo y 0.25 veces el largo libre si se trata de un extremo de la viga o 0.3 veces el largo libre de la viga si éste continua en otra viga y cruza una columna, a esto se le añade el desarrollo de su gancho o la mitad del ancho de la columna que cruza. Para cada suple sería:

$$\begin{aligned} \text{Costo suple 1} = & \text{MAX} (0.25l_o + l_{dh} + (6 + 12) \cdot db, l_d + l_{dh} + (6 + 12)db) \\ & \cdot \text{Área suple1} \cdot \$\text{acero} \end{aligned}$$

$$\text{Costo suple 2} = \text{MAX}(0.30l_o + bcol/2, l_d) \cdot \text{Área suple2} \cdot \$\text{acero}$$

Y el costo total de todas las vigas del piso analizado:

$$\text{Costo suples} = 2 \cdot (\text{Costo suple 1} + (N^{\circ} \text{Vigas} - 1) \cdot \text{Costo suple 2})$$

Refuerzo lateral:

El costo total de todas las vigas del piso analizado:

Costo laterales

$$= (2 \cdot (l_{dh} + (6 + 12) \cdot db) + N^{\circ}Vigas \cdot (l_o + l_d + bcol2) - bcol2) \\ \cdot \text{Área laterales} \cdot \$acero$$

Armadura Superior:

Costo armadura superior

$$= (2 \cdot (l_{dh} + (6 + 12) \cdot db) + N^{\circ}Vigas \cdot (l_o + l_d + bcol2) - bcol2) \\ \cdot \text{Área armadura superior} \cdot \$acero$$

Armadura inferior:

Costo armadura inferior

$$= (2 \cdot (l_{dh} + (6 + 12) \cdot db) + N^{\circ}Vigas \cdot (l_o + l_d + bcol2) - bcol2) \\ \cdot \text{Área armadura inferior} \cdot \$acero$$

8.8.3. Corrección de momentos

Se comparan los momentos nominales de las vigas con los momentos máximos últimos de entrada para las columnas, para cumplir con la ecuación 18.7.3.2 del código ACI 318S-14 (Comité ACI 318, 2014) (ecuación (8.1)), en la Figura 24 se muestra el esquema de los momentos.

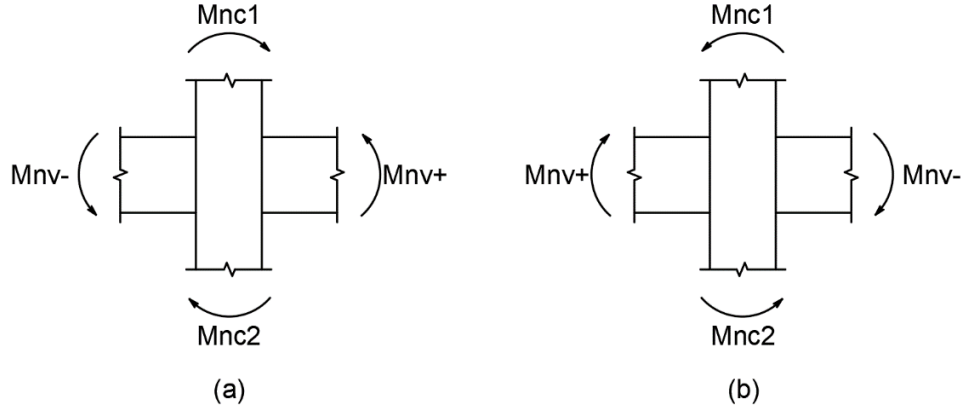


Figura 24: Criterio de sumatoria de momentos en un nodo para un diseño columna fuerte y viga débil.

$$M_{nc1} + M_{nc2} \geq 1.2(M_{nv+} + M_{nv-}) \quad (8.1)$$

Si este criterio no se cumple, entonces se calcula la diferencia de estos momentos:

$$Diferencia = 1.2 \sum M_{nv} - \sum M_{nc}$$

Finalmente, se corrige cada uno de los momentos últimos de las columnas, ajustando la diferencia de forma lineal de modo tal que los momentos últimos de entrada de las columnas cumplan el criterio de columna fuerte y viga débil, criterio que no aplica al último piso. Este ajuste se realiza de la siguiente forma:

$$M_{nc1} \text{ nuevo} = Diferencia \cdot \frac{M_{nc1}}{M_{nc1} + M_{nc2}} + M_{nc1}$$

$$M_{nc2} \text{ nuevo} = Diferencia \cdot \frac{M_{nc2}}{M_{nc1} + M_{nc2}} + M_{nc2}$$

8.8.4.Cálculo de columnas óptimas

En el cálculo de columnas óptimas se consideran dos tipos de columnas por piso, unas que van en los extremos y otras al centro, considerando para cada caso las solicitaciones mayores.

8.8.4.1. Cálculo de refuerzo longitudinal

Para el cálculo de la sección sometida a flexo-compresión, se encuentra en un comienzo el valor de la profundidad de la línea neutra 'c', donde la carga sea cero (flexión simple) o, de acuerdo con el extracto de curvas de interacción del anexo C.2. Para ello es necesario fijar los límites de 'c', siendo el mínimo 'c1' y el máximo 'c2', cuyos valores son:

$$c1 = 0$$

$$c2 = \text{MAX}\left(3h, d/\beta_1\right)$$

Luego, se define un valor intermedio a ser evaluado que corresponde a la semisuma de ambos:

$$ci = \frac{c1 + c2}{2}$$

Y se evalúa en las expresiones relacionadas al cálculo de carga nominal P_n , del anexo C.2, considerando que si el valor de P_n calculado es positivo, la solución está bajo el valor calculado de ci , por lo que se actualizan los valores de $c1$ y $c2$ como $c1 = c1$ y $c2 = ci$, en caso contrario, si el signo de P_n es negativo, los valores se actualizan a $c1 = ci$ y $c2 = c2$ y se actualiza el valor de ci a la semisuma de los nuevos $c1$ y $c2$, hasta encontrar un valor de 'c' cuyo resultado de P_n sea cero.

A continuación, se desea encontrar el valor de 'c' para que verifique el máximo valor absoluto de M_u y el valor de P_u , con una determinada excentricidad $e = \frac{M_u}{P_u}$, cuyos valores absolutos de P_u , inferiores a 0.2 [Tf], son considerados como nulos para evitar excentricidades muy grandes. Si el valor de P_u es positivo, se considera el valor del límite inferior $c1$ como el obtenido para una carga nula y $c2$ igual que en el caso anterior, en caso contrario, $c2$ toma el valor para c cuando la carga axial es nula y $c1$ será cero.

El método para encontrar el valor de c dada una combinación de momento y carga últimos es similar, con la diferencia que el criterio evaluado es la excentricidad $e = \frac{M_u}{P_u}$, y la diferencia de ésta respecto a la calculada $e_x = \frac{M_n}{P_n}$ para el valor de c evaluado debe ser menor a ± 0.001 .

Luego de obtener los parámetros anteriores, se puede calcular el factor de utilización, considerando para ello las cargas y momentos nominales minorizados por el factor de reducción correspondiente.

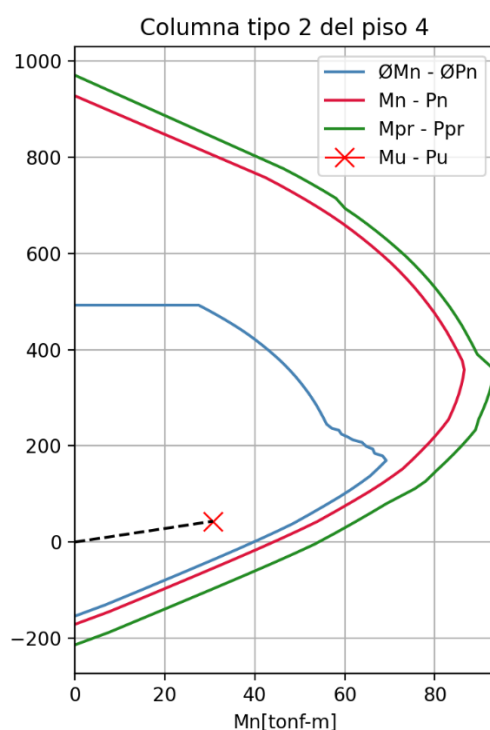


Figura 25: Curvas de interacción de una columna sometida a flexo-compresión.

Cabe mencionar que también se verifican las combinaciones de menores cargas y momentos para cubrir todo el espectro de solicitaciones. Con esto, se considera un factor de utilización para cada caso y se verifica que ambos cumplan con un valor inferior al 99.9%, por otro lado,

se evalúan las cuantías mínimas y máximas y que solo pasen los valores que sean capaces de cumplir todos criterios de diseño considerados en esta memoria.

En la Figura 25 se pueden visualizar las curvas de interacción, donde la curva azul corresponde a las cargas y momentos nominales minorizados, la curva roja a las cargas y momentos nominales, la curva verde a las cargas y momentos probables, y la x y su proyección, las cargas y momentos máximos solicitados.

El cálculo de refuerzo longitudinal para columnas sigue la misma línea que el refuerzo longitudinal para vigas, por lo que cualquier diseño de este tipo estará condicionado a que verifique los esfuerzos de corte.

Se omiten detalles acerca del cambio de sección, ya que se considera un costo relativamente menor, además de que estas varían en no más de 5 cm de ancho y largo, considerando las columnas inferiores de mayor tamaño o igual que las superiores.

Por otro lado, las barras de una misma sección se consideran de un mismo diámetro.

8.8.4.2. Cálculo de refuerzo transversal

Para el cálculo del refuerzo transversal se consideran 3 zonas de estribos, las zonas de rótulas plásticas, las zonas intermedias y la zona de empalme, esto se realiza de acuerdo con los criterios de diseño mencionados anteriormente, y se extenderán los estribos en la zona del nodo como si se tratara de un empalme.

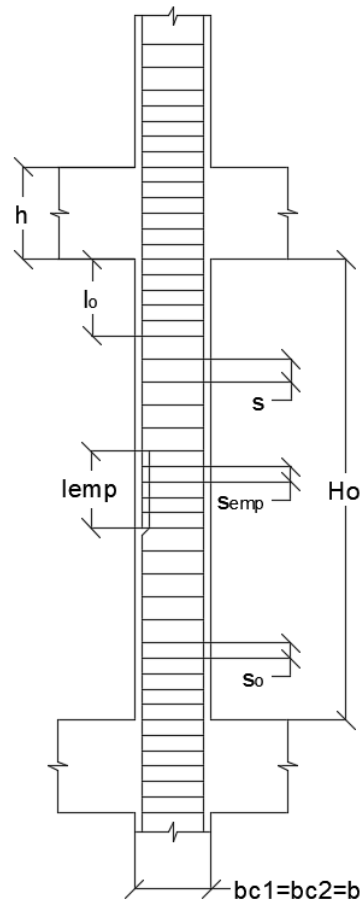


Figura 26: Detalle de columna en marco plano.

El criterio de verificación será similar al de las vigas, con la salvedad de que los cortes en las columnas tienden a ser de la misma magnitud en toda la sección y, por otro lado, se consideran dos diámetros por estribo, un diámetro para el estribo principal y el otro para los estribos interiores, siendo este último menor o igual al primero. Esto se puede visualizar mejor en la Figura 27, donde el estribo rojo representa el estribo exterior y el verde el interior.

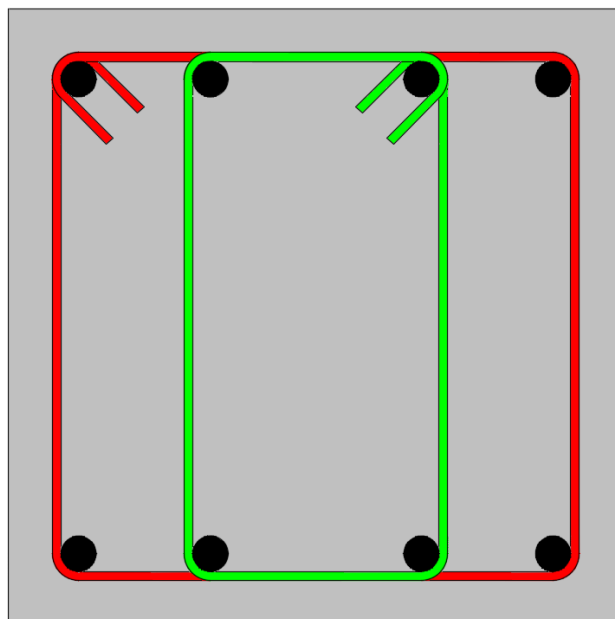


Figura 27: Estribos interiores y exteriores.

La función de costos para este caso es:

$$\text{Costo estribos} = \text{costo 1} + \text{costo 2} + \text{costo 3} + \text{costo 4}$$

Donde el costo 1 corresponde a las zonas de rótula plástica, el costo 2 a la zona libre, el costo 3 a la zona de empalme y el costo 4 a la zona del nudo. Lo anterior se describe como se muestra a continuación:

Costo 1:

$$\text{Costo 1} = 2 \cdot N^{\circ} \text{estribos en zona de RP} \cdot (\text{largo total exterior 1} \cdot \text{Área exterior 1} + 2 \cdot \text{largo total interior 1} \cdot \text{Área interior 1}) \cdot \$\text{acero}$$

Costo 2:

$$\text{Costo 2} = N^{\circ} \text{estribos en zona libre} \cdot (\text{largo total exterior 2} \cdot \text{Área exterior 2} + 2 \cdot \text{largo total interior 2} \cdot \text{Área interior 2}) \cdot \$\text{acero}$$

Costo3:

$$\text{Costo3} = N^{\circ}\text{estribos en zona de empalme} \cdot (\text{largo total exterior 3} \cdot \text{Área exterior 3} + 2 \cdot \text{largo total interior 3} \cdot \text{Área interior 3}) \cdot \$\text{acero}$$

Costo 4:

$$\text{Costo 4} = N^{\circ}\text{estribos en zona de nudo} \cdot (\text{largo total exterior 4} \cdot \text{Área exterior 4} + 2 \cdot \text{largo total interior 4} \cdot \text{Área interior 4}) \cdot \$\text{acero}$$

Siendo:

largo total exterior n: el largo del estribo exterior del estribo *n*

Área exterior n: el área del diámetro exterior del estribo *n*

largo total einterior n: el largo del estribo interior del estribo *n*

Área exterior n: el área del diámetro interior del estribo *n*

Y *n* el número de estribo según la zona que se esté evaluando.

8.8.4.3. Función de costo para las columnas de cada piso

Luego de que verifique la combinación de elementos para la armadura longitudinal y de corte junto a su costo, se calcula el costo total de las columnas del mismo tipo analizadas, considerando que para las barras longitudinales existe solo un empalme central, considerando la longitud de todas las barras hasta ese punto independiente de si existe una reducción en el número de éstas. La función de costos por tipo de columnas en cada piso es:

$$\begin{aligned} & \text{Largo de barras longitudinales} \\ & = \text{Largo empalme} + \text{Altura libre} + \text{Largo desarrollo} \end{aligned}$$

$$\sum \text{Costo Columnas } n = N^{\circ} \text{ columnas tipo } n \cdot$$

$$(\text{Largo de barras longitudinales} \cdot \text{Área barras longitudinales} \cdot N^{\circ} \text{ barras} \cdot \$\text{acero} \\ + (\text{altura libre} + \text{altura viga}) \cdot b_{c1} \cdot b_{c2} + \text{Costo estribos})$$

8.8.5. Función de costos de toda la estructura

Costo total estructura por piso

$$= \sum \text{costo columnas } 1 + \sum \text{costo columnas } 2 + \sum \text{costo vigas}$$

$$\text{Costo total estructura} = \sum \text{Costo total estructura por piso}$$

9. Conclusiones y Recomendaciones

Se logró desarrollar un software de código abierto, el cual replica una funcionalidad esencial para cualquier programa de análisis estructural, como lo es el poder calcular los esfuerzos internos de una estructura, en este caso, aplicado a marcos resistentes a momento.

Si bien, el programa en principio es simple, permite sentar las bases para poder diseñar una interfaz gráfica, la que podría usar el resto de las funcionalidades programadas de manera sencilla, como la adición y edición de elementos, haciendo de β -beam un programa casi tan completo como la versión demo del RISA-3D, ignorando el diseño de perfiles metálicos, diferenciados solamente por el enfoque en cuanto a diseño de secciones y que, con β -beam, es posible guardar los trabajos realizados.

El programa desarrollado se ajusta a los requerimientos estipulados, ya que es un programa liviano, de fácil instalación y manipulación. Entrega una solución económica para el prediseño de marcos especiales resistentes a momento, de forma rápida y con una muy buena descripción de la disposición de las barras de acero longitudinal, sus estribos y las secciones de hormigón, además realiza la cubicación de los materiales junto al costo total y muestra el factor de utilización de los elementos, todo lo anterior en un proceso completamente automático y con datos de entrada simples.

La tendencia del programa es elegir la sección más grande de hormigón mientras su cuantía sea cercana al mínimo exigido, por lo que muchas veces, al limitar la altura de una viga, por ejemplo, a 10 o 15 centímetros menos, la sección resulta más cara de lo que cuesta con esos 10 o 15 centímetros que se le restaron, esto porque la armadura longitudinal en vigas es mayor por lo general, porque el diseño controla por flexión, no así una columna, cuyos esfuerzos de corte son mayores.

Para el diseño de columnas, no siempre los valores máximos de carga y momento son los que controlan a la sección, sino que uno donde exista una gran excentricidad, por esto se evaluaron ambos casos.

El hecho de detallar los elementos genera una mejor percepción de factores a cuidar como uniones, dobleces, traslapes, distancias de separación y posibles encuentros complicados de barras.

En cuanto a la experiencia programando, muchos de los métodos utilizados en papel tienen que ser reordenados y planteados de una forma en la que se reduzca la mayor cantidad de pasos en cada uno de los cálculos. Esto se da por la manera en que el programar le obliga a uno a plantear todo linealmente sin caer en redundancias, y esto puede aplicarse a situaciones donde no necesariamente se desarrolla software, sino que simple álgebra o en la planificación de otros proyectos de ingeniería.

Para mejorar el programa, se sugiere empezar por añadir las siguientes características: expandir el análisis matricial de estructuras a 3 dimensiones y crear una interfaz gráfica para hacer más grata la experiencia de usuario. Estas son características esenciales para tener un programa completo que pueda empezar a competir con otros más allá del ámbito académico, sino que también en el ámbito laboral

Se propone a las escuelas de ingeniería crear un repositorio con varias herramientas de distintas especialidades para la resolución de distintos problemas, que integren a sus alumnos y profesores, permitiéndoles participar e ir mejorando, añadiendo funcionalidad a los proyectos que se puedan generar, esto para facilitar la adquisición de nuevos conocimientos y poniéndolos a prueba antes de que los alumnos se enfrenten al mundo laboral, ya que el desarrollo de un programa computacional es equivalente a enseñarle a una máquina a resolver un problema.

Referencias

- ACI Committee. (2005). *Building code requirements for structural concrete (ACI 318-05) and commentary (ACI 318R-05)*. American Concrete Institute.
- Araya-Castillo, L. y Pedreros-Gajardo, M. (2013). *PIRATERÍA DE SOFTWARE: PROPUESTA DE MODELO*. Dialnet. Obtenido de <https://dialnet.unirioja.es/descarga/articulo/4692054.pdf>
- Autodesk. (s.f.) *What Is BIM / Building Information Modeling*. Autodesk. Recuperado el 9 de marzo de 2021, de <https://www.autodesk.com/solutions/bim>
- Belmont, L., Smith, B., y Vink, R. (2021). *anaStruct*. GitHub. Obtenido de <https://github.com/ritchie46/anaStruct>
- Bridgwater, A. (16 de septiembre de 2020). *Why Do We Still Need More Software?*. Forbes. Obtenido de <https://www.forbes.com/sites/adrianbridgwater/2020/09/16/why-do-we-still-need-more-software/?sh=62ae10285e54>
- Business Software Alliance. (2018). *Gestión de software: obligación de seguridad, oportunidad de negocios*.
- Civil Engineering Software Database. (s.f.) *DELTA BEAM*. Civil Engineering Software Database. Recuperado el 9 de marzo de 2021, de <https://www.cesdb.com/delta-beam.html>
- Comité ACI 318. (2014). *Requisitos de Reglamento para Concreto Estructural (ACI 318S-14) y Comentario (ACI 318SR-14)*. American Concrete Institute, USA.
- Cooperativa. (3 de enero de 2016). *Más de 1300 procesos por uso ilegal de software al cierre de 2015 en Chile*. Cooperativa. Obtenido de

<https://www.cooperativa.cl/noticias/tecnologia/industria/informatica/mas-de-1300-procesos-por-uso-ilegal-de-software-al-cierre-de-2015-en-chile/2016-01-03/181539.html>

Cross, H. (1930). *Analysis of Continuous Frames by Distributing Fixed-End Moments*. Proceedings of the American Society of Civil Engineers. ASCE. pp. 919–928.

Díaz, M. y Massa, J. (s.f.). *W-Trite*. Recuperado el 9 de marzo de 2021, de https://web.archive.org/web/20090828033827/http://wtrite.net/a_cerca.html

Drummond, C. (27 de octubre de 2017). *Las profesiones más demandadas del mundo actualmente*. TICbeat. Obtenido de <https://www.ticbeat.com/empresa-b2b/estas-son-las-profesiones-mas-demandadas-del-mundo-actualmente/>

Fanjul, S. (23 de junio de 2019). *Las carreras más demandadas en un mundo cambiante*. EL PAÍS. Obtenido de https://elpais.com/elpais/2019/06/20/actualidad/1561042048_006142.html

Gavin, H. (2014). *Frame3DD*. Sourceforge. Obtenido de <http://frame3dd.sourceforge.net/>

Grand Canyon University. (25 de septiembre de 2020). *Why Is Programming Important?*. Obtenido de <https://www.gcu.edu/blog/engineering-technology/computer-programming-importance>

Guendelman, T. (2014). *Análisis estático y dinámico de estructuras*. La Serena, Editorial Universidad de La Serena.

Gupta, N. y Jamwal, S. (2012). *Software Piracy among IT students of J&K: Ethical or Unethical*. International Journal of Computer Applications. Obtenido de https://www.academia.edu/1585893/Software_Piracy_among_IT_students_of_J_and_K_Ethical_or_Unethical

- Hao, K. (10 de noviembre de 2020). *La IA ha resuelto un acertijo matemático clave para entender el mundo*. MIT Technology Review. Obtenido de <https://www.technologyreview.es/s/12810/la-ia-ha-resuelto-un-acertijo-matematico-clave-para-entender-el-mundo>
- Heaven, W. (30 de noviembre de 2020). *DeepMind's protein-folding AI has solved a 50-year-old grand challenge of biology*. MIT Technology Review. Obtenido de <https://www.technologyreview.com/2020/11/30/1012712/deepmind-protein-folding-ai-solved-biology-science-drugs-disease/>
- Iran University of Science and Technology. (s.f.). *Structural Mechanics; Graph and Matrix Methods. Chapter 6: "Matrix Force Method"*. Recuperado el 7 de enero de 2021, de http://www.iust.ac.ir/files/cefsse/pg.cef/Contents/force_method_ch6.pdf
- Kini, R. (2008). *Software Piracy in Chilean e-Society*. The International Federation for Information Processing, vol 286. Springer, Boston, MA. doi:[10.1007/978-0-387-85691-9_25](https://doi.org/10.1007/978-0-387-85691-9_25)
- Maney, George A. (1915). *Studies in Engineering*. Minneapolis: University of Minnesota.
- Microjuris. (10 de abril de 2009). *Corte de Santiago acoge demanda por uso sin licencia de software*. Microjuris. Obtenido de <https://aldiachile.microjuris.com/2019/04/10/corte-de-santiago-acoge-demanda-por-uso-sin-licencia-de-software/>
- Nair, R. V. (2013). *Performance Assessment of Multi-storeyed RC Special Moment Resisting Frames (Doctoral dissertation)*. National Institute of Technology. India, Rourkela.
- NCh 1537. (2009). *Cargas permanentes y sobrecargas de uso*. Instituto Nacional de Normalización, Santiago, Chile.

- NCh 3171. (2010). *Diseño estructural - Disposiciones generales y combinaciones de carga*. Instituto Nacional de Normalización, Santiago, Chile.
- NCh 433. (1996). *Diseño Sísmico de edificios, incluyendo las modificaciones del año 2012 y Decreto Supremo 61*. Instituto Nacional de Normalización, Santiago, Chile.
- Nassani, D. (diciembre de 2013). *A Simple Model for Calculating the Fundamental Period of Vibration in Steel Structures*. Turquía: Elsevier B. V. doi:[10.1016/j.apcbee.2014.01.060](https://doi.org/10.1016/j.apcbee.2014.01.060)
- Nieves, J. (29 de diciembre de 2020). *Una IA consigue resolver la ecuación de Schrödinger*. Diario ABC. Obtenido de https://www.abc.es/ciencia/abci-consigue-resolver-ecuacion-schrodinger-202012292032_noticia.html
- Nilson, A. H. y Darwin, D. (1999). *Diseño de estructuras de concreto*. Colombia, McGraw-Hill.
- Park, R. y Paulay, T. (1983). *Estructuras de Concreto Reforzado*. México. D. F.: Editorial Limusa.
- Petrescu, M. (23 de junio de 2013). *bars3d*. Sourceforge. Obtenido de <https://sourceforge.net/projects/bars3d/>
- Petro, Y. (29 de mayo de 2014). *DS-Frame2D*. Sourceforge. Obtenido de <https://sourceforge.net/projects/dsframe2d/>
- Rodríguez, M. (5 de septiembre de 2019). *Las 10 profesiones con más demanda en EE.UU.* ThoughtCo. Obtenido de <https://www.thoughtco.com/profesiones-con-mayor-futuro-en-eeuu-1965575>
- Setareh, M., y Darvas, R. (2017). *Concrete structures*. Suiza: Springer. doi:[10.1007/978-3-319-24115-9](https://doi.org/10.1007/978-3-319-24115-9)

Sourceforge. (28 de febrero de 2015). *PSA*. Sourceforge. Obtenido de <https://sourceforge.net/projects/psafem/>

Thakkar, B. (3 de septiembre de 2018). *Modal Analysis of Plane Truss using Python*. Code Project. Obtenido de <https://www.codeproject.com/Articles/1258509/Modal-Analysis-of-Plane-Truss-using-Python>

The Free Encyclopedia. (3 de diciembre de 2020). *List of structural engineering software*. PIR. Recuperado el 9 de marzo de 2021, de <https://tinyurl.com/orubapkcuf>

Vallejos, W. (s.f.). *La amenaza del negocio software: Los piratas de hoy*. Universidad de Chile. Recuperado el 9 de marzo de 2021, de: <http://www.periodismo.uchile.cl/themoroso/2001/internet/pirateria.htm>

Walsink. (s.f.). *FRAMEWORK 2D+3D*. Recuperado el 9 de marzo de 2021, de <https://members.ziggo.nl/wolsink/>

Bibliografía

- [1] Llorca, Á. (12 de enero de 2017). *Qué son las botnets y por qué son un peligro creciente*. Obtenido de <https://www.genbeta.com/seguridad/que-son-las-botnets-y-por-que-se-usaron-en-el-ataque-ddos-del-viernes>
- [2] Python Foundation. (2021). *Python 3.9.4 documentation*. Obtenido de <https://docs.python.org/3.9/>
- En esta documentación se encuentra disponible toda la información necesaria para aprender a usar el lenguaje de programación Python. Se recomienda la lectura de las secciones “Tutorial/An Informal Introduction to Python”, “Tutorial/More Control Flow Tools” y “Tutorial/Data Structures”, las cuales son suficientes para llevar una lectura fluida de la mayoría del código fuente del programa β -beam.*

Anexo A – Repositorios

En este anexo se mencionan los sitios donde se encuentra almacenado el código fuente del programa β -beam para su libre distribución y como usarlos para poder descargarlo. Cabe destacar que, para cualquier método, es necesario tener previamente instalado Python en su última versión para poder ejecutarlo.

A.1. Repositorio GitHub

GitHub es una de las plataformas más utilizadas para compartir y distribuir proyectos de desarrollo de software.

El repositorio del programa, con el código fuente y sus requisitos, se encuentra en el siguiente enlace:

<https://github.com/patod01/beta-beam>

Descargando el proyecto, se encuentran los archivos esenciales en el directorio “src”. Sus dependencias deberán ser descargadas manualmente como se hace normalmente con los módulos de Python. Otra opción es utilizar la herramienta “pip”, la cual instala todos los requisitos automáticamente.

A.2. Repositorio PyPi

PyPi es la plataforma oficial para la distribución de módulos de Python por parte de la comunidad. Para instalar β -beam desde PyPi, solo se debe escribir en la línea de comandos la siguiente instrucción:

- `pip install beta-beam`

Anexo B – Demostración

En este anexo se detalla un enunciado de ejemplo en el ítem B.0, para el cual se definen los datos de entrada a β -beam en el ítem B.1. Luego se comprueban los esfuerzos internos de un marco sometido a cargas vivas, muertas y sísmicas, dadas las condiciones iniciales de un proyecto, contrastando con el mismo marco modelado en el programa RISA-3D (versión demo) en el ítem B.2. Posteriormente, en el ítem B.3, se indican las iteraciones hasta encontrar un conjunto de secciones que se aproximen a cumplir con el drift máximo, para luego comenzar con la etapa de diseño de vigas y columnas, en el ítem B.4, mostrando el primer output de estas rutinas. Finalmente, en el ítem B.5., se muestran los valores finales a los que llegó el programa, indicando el número de iteraciones, siendo estos valores la propuesta para un prediseño económico.

B.0. Enunciado del proyecto

Para la demostración del programa, se define el siguiente proyecto:

Calcular una estructura de marcos resistentes a momento que cuente con 4 pisos y 3 vanos por marco, teniendo la estructura un total de 4 marcos distanciados 7 metros entre sí. La altura entre piso es de 3 metros y cada vano tiene un ancho de 7 metros. Se requieren espesores de losa de 15 centímetros. La estructura es doblemente simétrica, por lo que se desprecia cualquier efecto producto de una torsión accidental. La estructura se proyecta en un suelo tipo E, zona sísmica 3 y la edificación está destinada al uso habitacional, estimándose una sobrecarga para todos los pisos de la estructura de 500 kg/m^2 .

B.1. Configuración del proyecto

Inicialmente se ejecuta el programa para generar el espacio de trabajo con los valores por defecto. En el archivo *input.txt* generado se agregan las condiciones del enunciado teniendo en cuenta las unidades que exige el programa para el ingreso de los datos de entrada. Se dejan por defecto las secciones iniciales, el módulo de *Poisson* y el resto se rellena manualmente.

Se verifican las condiciones para usar el método estático en el ítem 6.2.1 de la norma NCh 433 (1996). La estructura cumple con el criterio b), no más de 5 pisos y menos de 20 metros de altura. Se procede a llenar el archivo con los datos de entrada, el que se ve como se presenta a continuación:

```

File Edit Format View Help
numero de pisos      : 4
numero de vanos      : 3
numero de marcos     : 4
altura de piso       : 300 [cm]
distancia entre columnas : 700 [cm]
distancia entre marcos : 700 [cm]

ancho de columnas    : 90 [cm]
ancho de vigas       : 40 [cm]
altura de vigas      : 90 [cm]
espesor de losas     : 15 [cm]

modulo de poisson v  : 0.25
resistencia f'c hormigon : 25 [MPa]
densidad del hormigon  : 2500 [kg/m3]
precio hormigon       : 75000 [$/m3]
precio acero          : 1000 [$/kg]

zona sismica         : 3
tipo de suelo        : E
categoria de edificacion : II
factor de modificacion R : 7
sobrecarga de uso     : 500 [kg/m2]
porcentaje de sobrecarga : 0.25

80% Windows (CRLF) UTF-8

```

Figura 28: Datos de entrada, marco de prueba.

Hay que tener presente que β -beam realiza un proceso iterativo previo a dar los resultados finales para presentar al usuario, por lo que los datos que se exponen a continuación son datos intermedios extraídos durante el tiempo de ejecución del código intervenido en un IDE (*integrated development environment*).

B.2. Comprobación de esfuerzos

Para contrastar algunos resultados respecto de los obtenidos con β -beam, se utilizó el programa en su versión demo, RISA-3D. El modelo se ve a continuación, considerando las secciones iniciales en base a las dimensiones máximas establecidas en los datos de entrada (vigas de 40 cm por 90 cm y columnas de 90 cm):

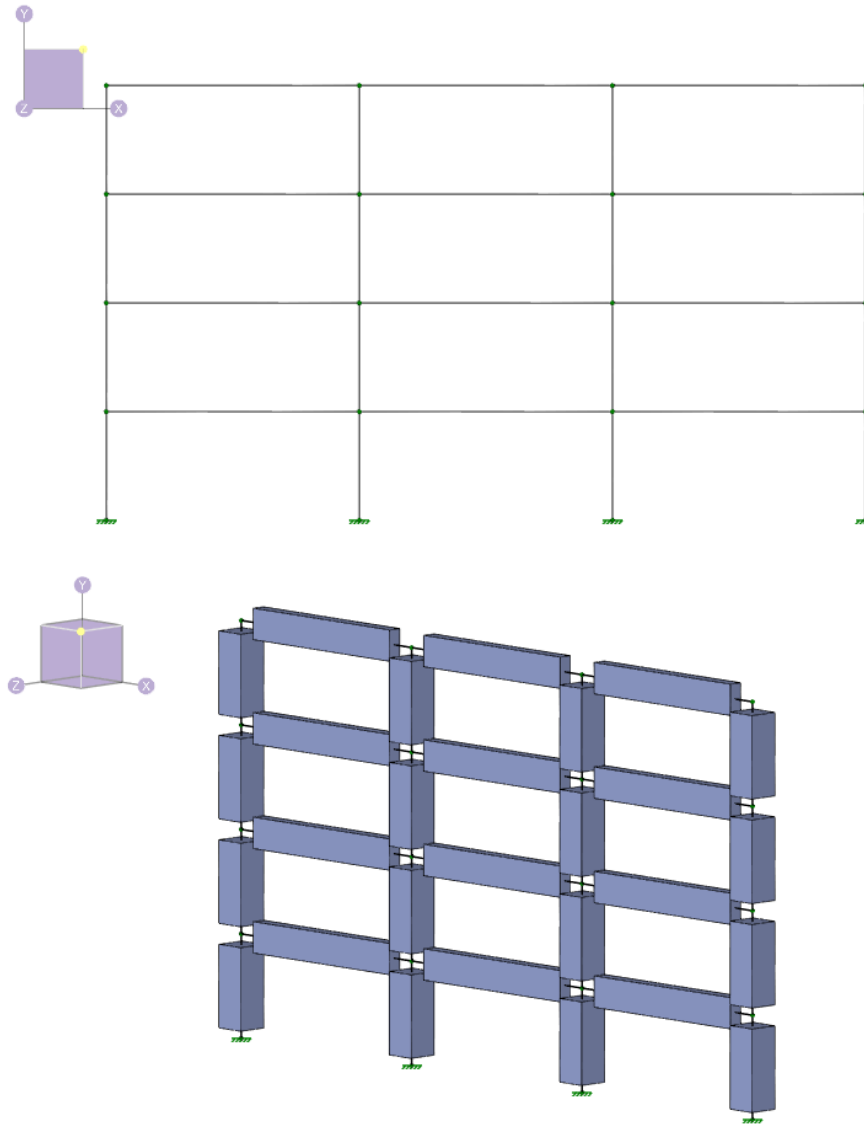


Figura 29: Modelo de marco inicial, RISA-3D.

Las cargas por categoría se presentan a continuación:

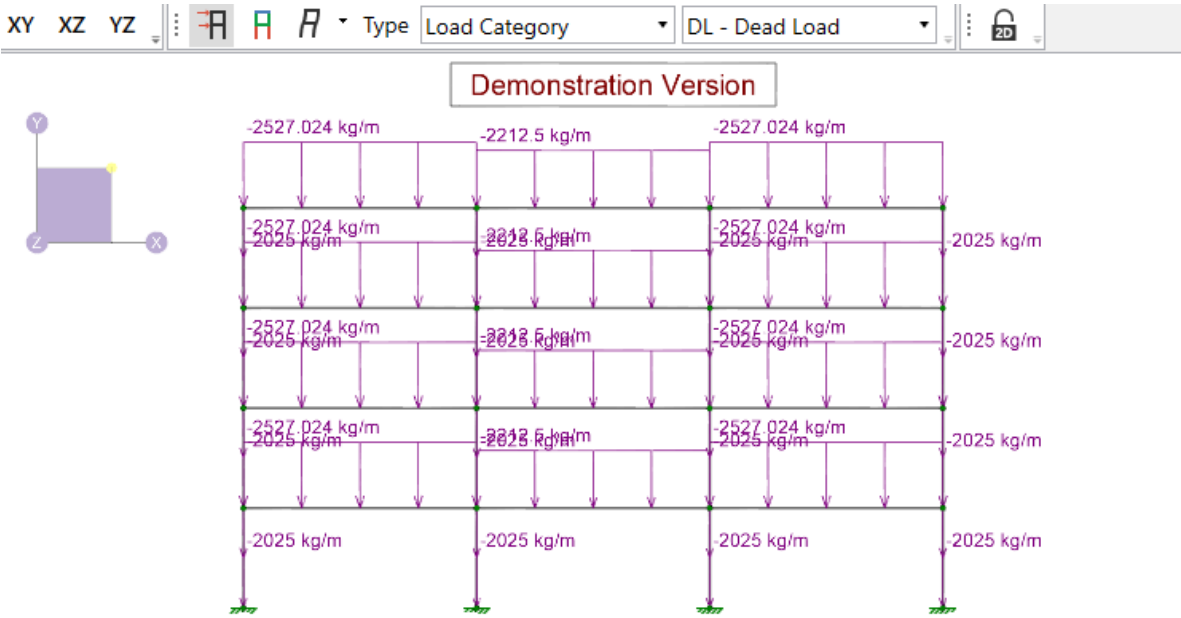


Figura 30: Cargas muertas totales, RISA-3D.

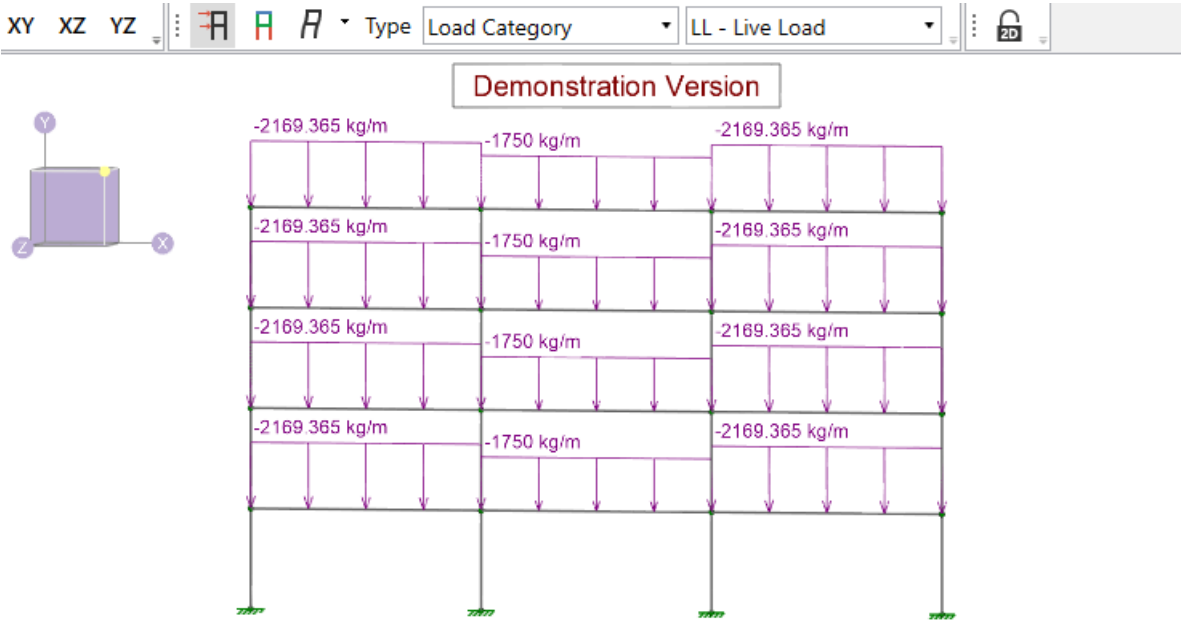


Figura 31: Cargas vivas totales, RISA-3D.

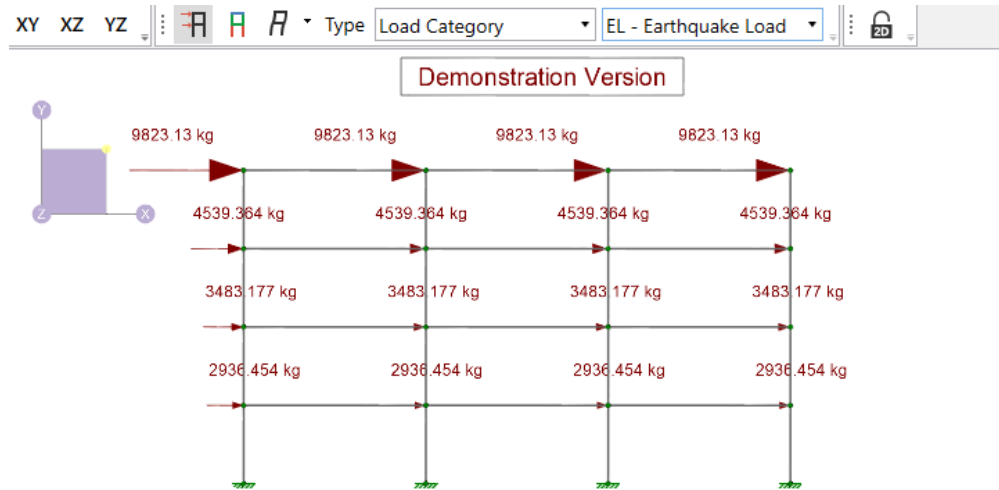


Figura 32: Fuerzas sísmicas, RISA-3D.

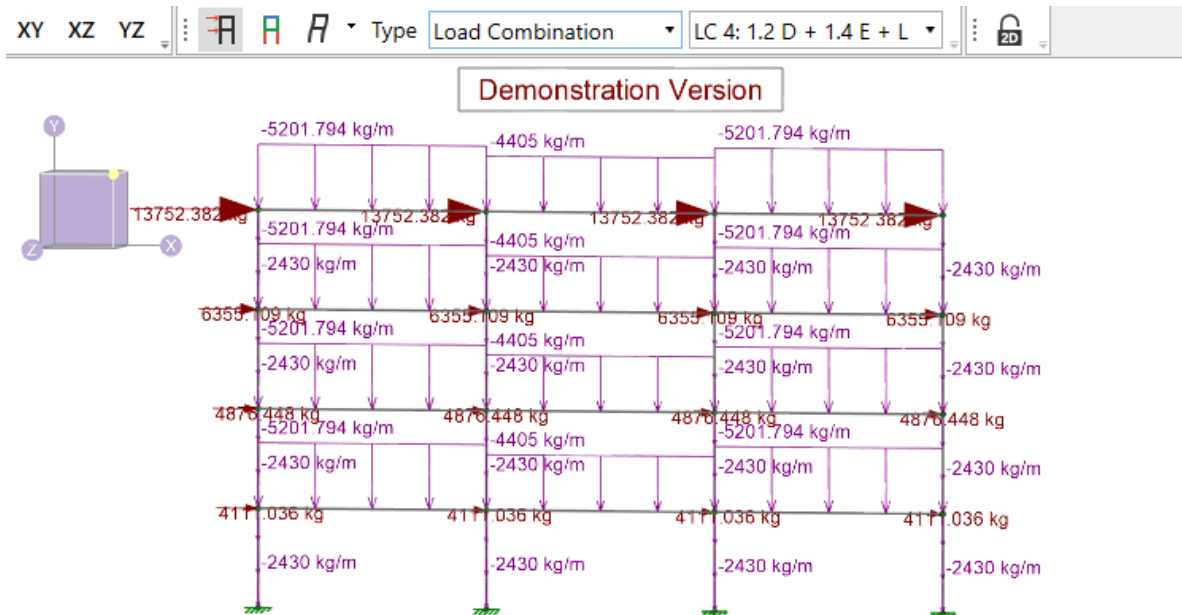


Figura 33: Combinación 1.2 D + 1.4 E + L, RISA-3D.

A continuación, se presentan tablas que muestran los resultados de los primeros 3 miembros, correspondientes a las 3 primeras columnas, de izquierda a derecha, del primer nivel, para la combinación $1.2 D + 1.4 E + L$, contrastando los resultados de RISA-3D y β -beam respectivamente:

Member Section Forces (By Combination)									
Sections	Maximums	End Reactions							
	LC	Member Label	Sec	Axial[kg]	y Shear[kg]	z Shear[kg]	Torque[...]	y-y Mo...	z-z Moment[kg-m]
1	4	M1	1	65295.982	20093.543	0	0	0	65400.562
2			2	58005.982	20093.543	0	0	0	5119.933
3	4	M2	1	165256.614	33401.901	0	0	0	79664.252
4			2	157966.614	33401.901	0	0	0	-20541.451
5	4	M3	1	162753.592	32458.276	0	0	0	78990.384
6			2	155463.592	32458.276	0	0	0	-18384.444

Figura 34: Esfuerzos internos, RISA-3D.

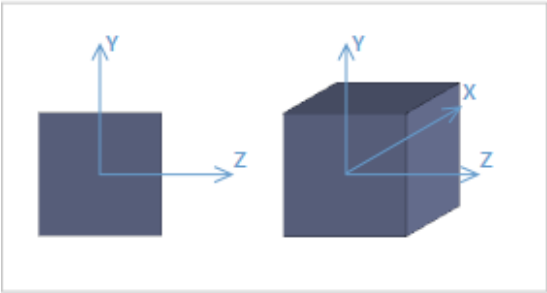
*DEL.txt - Notepad							
File Edit Format View Help							
fila	perfil	combo	nodo	[kg] axial	[kg] corte	[kg-m] momento	
7	COL 1	1.2 D + 1.4 E + L	(i)	65295.97	20093.54	65400.56	
8			(j)	58005.97	20093.54	5119.93	
21	COL 2	1.2 D + 1.4 E + L	(i)	165256.6	33401.9	79664.25	
22			(j)	157966.6	33401.9	-20541.45	
35	COL 3	1.2 D + 1.4 E + L	(i)	162753.58	32458.28	78990.38	
36			(j)	155463.58	32458.28	-18384.44	
Ln 547, Col 1				100%	Windows (CRLF)	UTF-8	

Figura 35: Esfuerzos internos, β -beam.

A continuación, se presentan parte de los respectivos informes de cada programa, donde se muestran los diagramas de esfuerzo interno del miembro 2 (columna) y miembro 18 (viga):

Detail Report: M2

Unity Check: 0.49 (shear)



Input Data:

Shape:	CRECT90X90
Member Type:	Column
Length (cm):	300
Material Type:	Concrete
Design Rule:	Typical
Number of Internal Sections:	40
Design Code:	ACI 318-05

Diagrams:

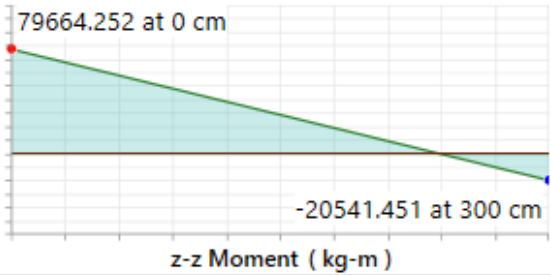
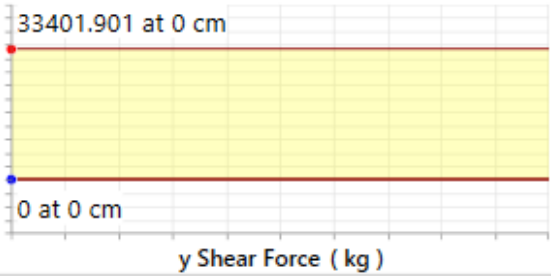
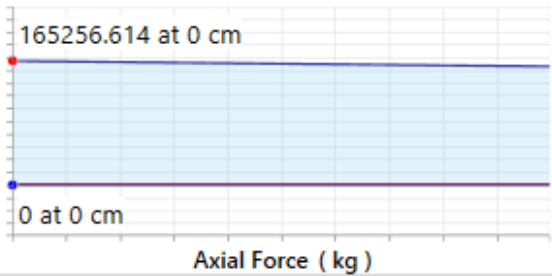
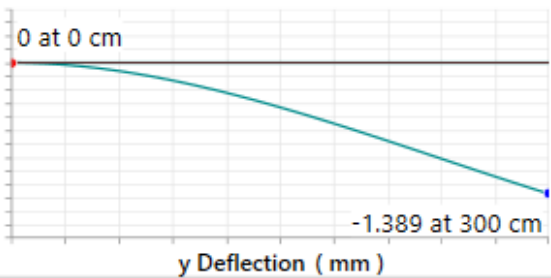


Figura 36: Reporte miembro 2, columna, RISA-3D.

Miembro 2: COL

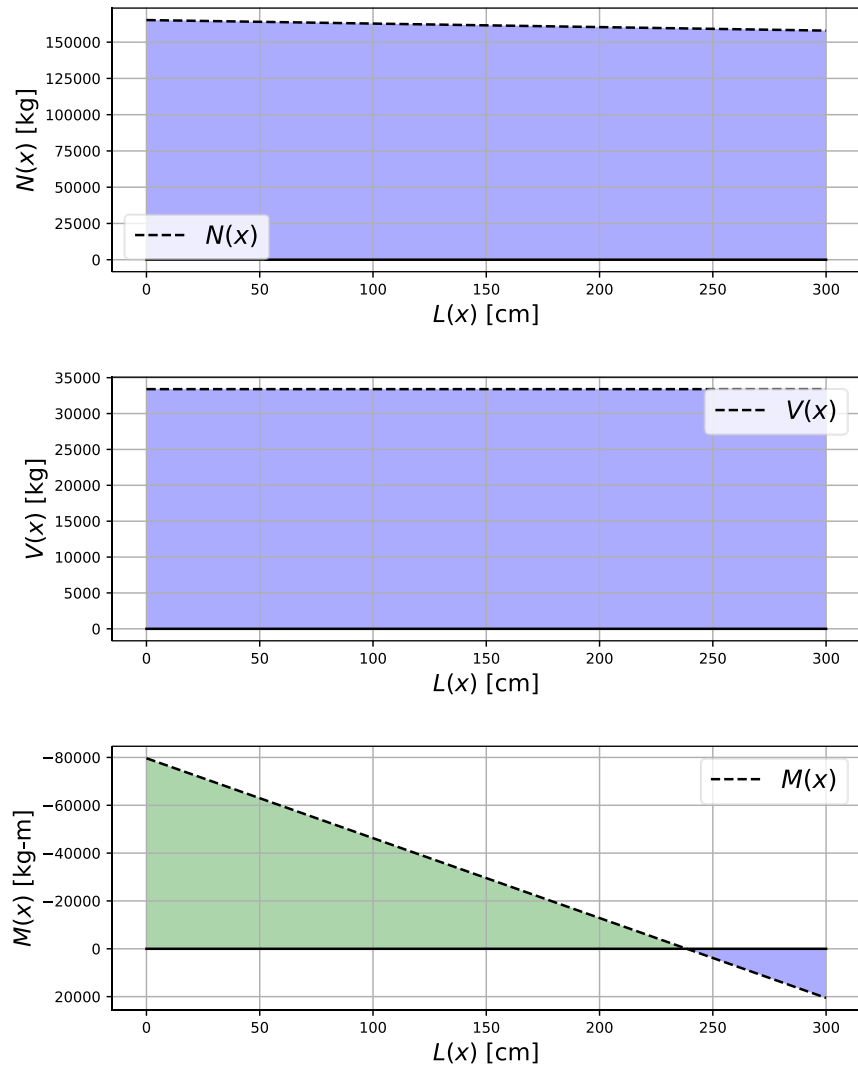
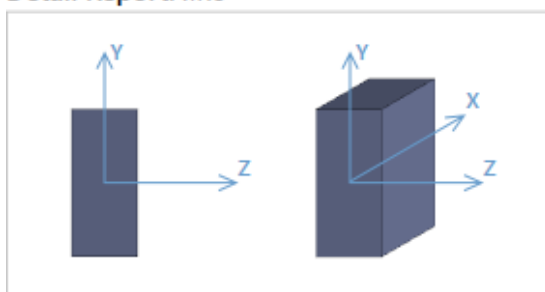


Figura 37: Reporte miembro 2, columna, β -beam.

Detail Report: M18

Unity Check: 0.855 (axial/bending)



Input Data:

Shape:	CRECT90X40
Member Type:	Beam
Length (cm):	700
Material Type:	Concrete
Design Rule:	Typical
Number of Internal Sections:	40
Design Code:	ACI 318-05

Diagrams:

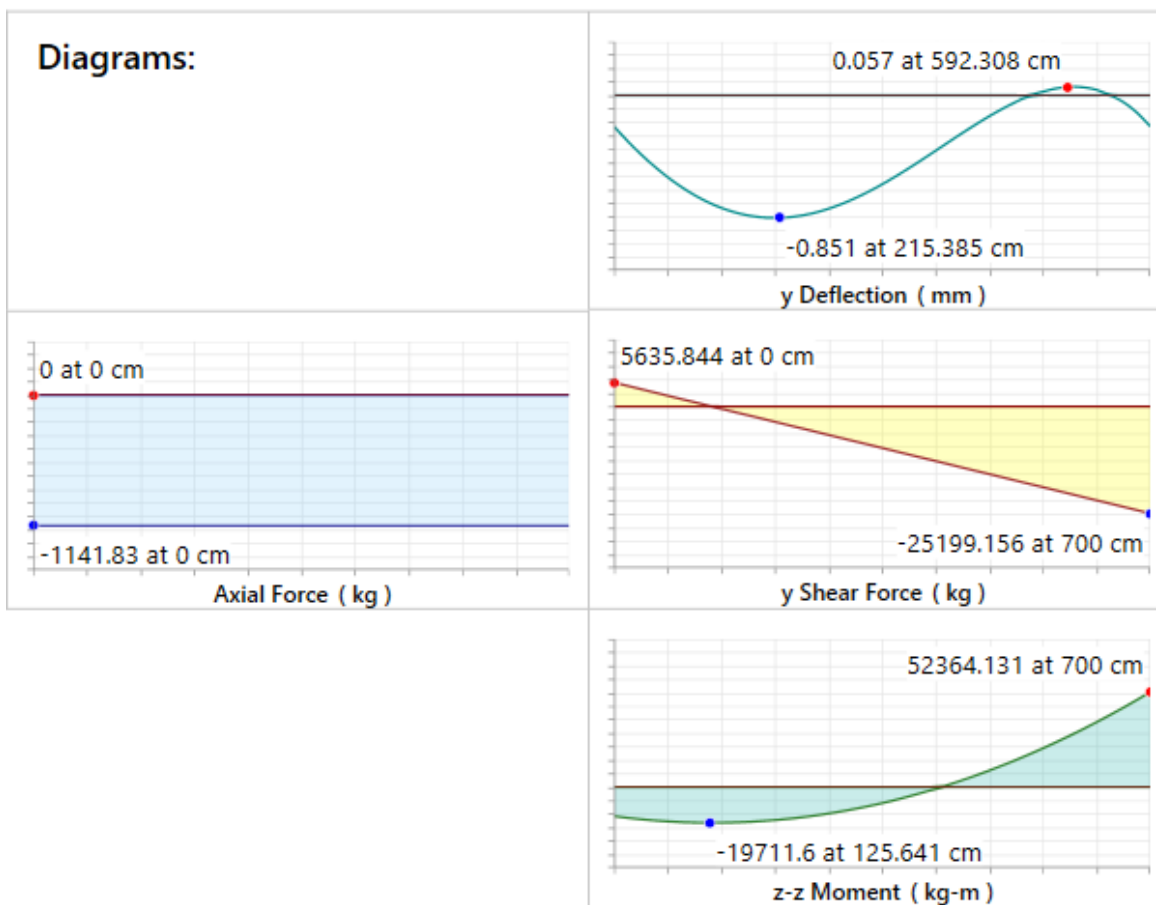


Figura 38: Reporte miembro 18, viga, RISA-3D.

Extremo i:
 $N(45.0): -1141.83 \text{ [kg]}$
 $V(45.0): 3653.59 \text{ [kg]}$
 $M(45.0): 18197.58 \text{ [kg-m]}$
 Extremo j:
 $N(655.0): -1141.83 \text{ [kg]}$
 $V(655.0): -23216.91 \text{ [kg]}$
 $M(655.0): -41470.54 \text{ [kg-m]}$

Miembro 18: VIGA

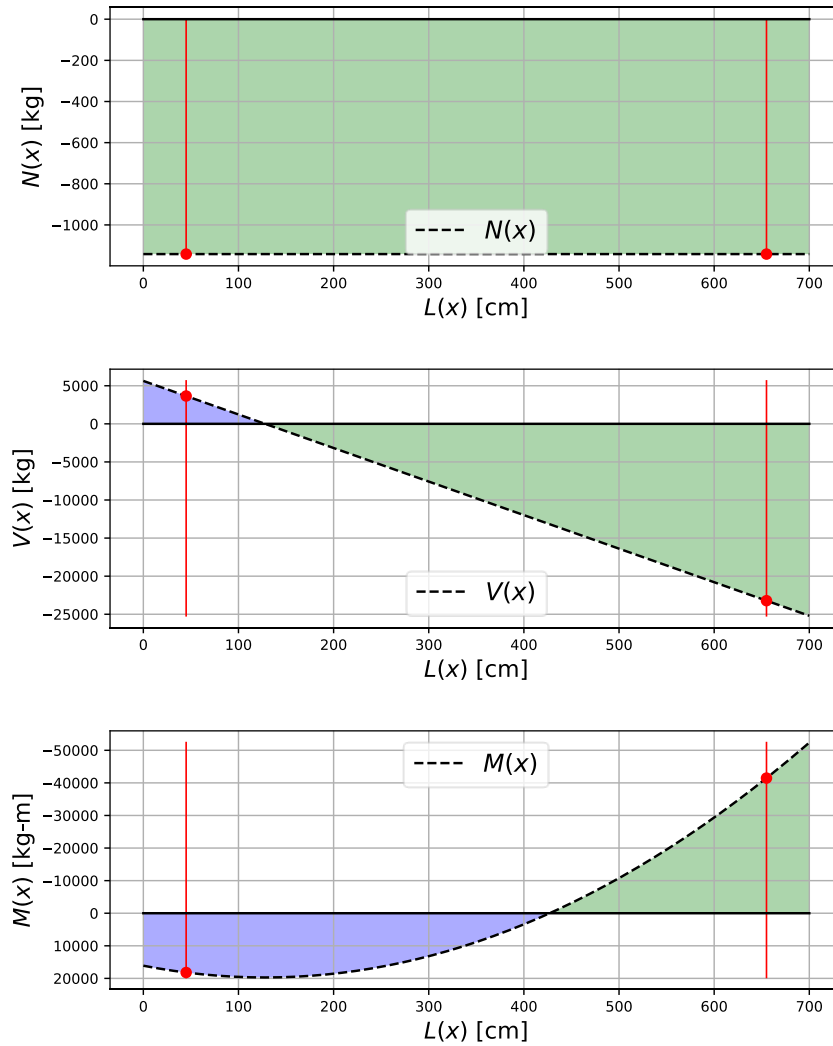


Figura 39: Reporte miembro 18, viga, β -beam.

Se puede comprobar que los resultados son prácticamente idénticos, lo que cumple uno de los objetivos fundamentales. En esta etapa se demuestra que, para propósitos genéricos y

pasando por alto el inconveniente de no tener una interfaz gráfica, β -beam es un programa funcional y resuelve estructuras hiperestáticas. En los próximos capítulos de este anexo se muestran los resultados tras las iteraciones de las herramientas de diseño con las que cuenta.

B.3. Verificación del drift máximo

En este capítulo se empieza con la optimización del marco, en primera instancia, acercándose al drift máximo para impedir tener una estructura excesivamente rígida. Los resultados del drift calculados por β -beam se presentan en la siguiente imagen:

fila	combo	nivel	[mm] δ_{max}	[mm] δ_{min}	[‰] drift
1	E	(01)	0.9953	0.9839	0.332
2	E	(02)	2.6835	2.6818	0.567
3	E	(03)	4.2595	4.2574	0.526
4	E	(04)	5.4373	5.4103	0.393
5	- E	(01)	-0.9953	-0.9839	-0.332
6	- E	(02)	-2.6835	-2.6818	-0.567
7	- E	(03)	-4.2595	-4.2574	-0.526
8	- E	(04)	-5.4373	-5.4103	-0.393

Figura 40: Desplazamientos mínimos y máximos por nivel, β -beam.

Por defecto, se calculan para ambas direcciones del sismo, pero dada la simetría de la estructura, con analizar una sola dirección es suficiente. Esto se toma en cuenta para iteraciones posteriores.

β -beam indica que el drift máximo es 0.567‰, para lo que sugiere un primer factor de corrección de área, $\psi = 0.7637$. Los valores presentados a continuación corresponden a parte de la interfaz para monitoreo del programa y muestra las secciones utilizadas, el tipo

de elemento y, en la zona de abajo, una lista con las nuevas secciones recomendadas, siendo estas la multiplicación del factor con cada una de las secciones anteriores:

```
drift maximo: 0.567  
ψ: 0.7637189685065491  
-----  
se necesita corregir los perfiles...  
-----  
90 90 COL  
90 90 COL  
90 90 COL  
90 90 COL  
90 90 COL  
90 90 COL  
90 90 COL  
90 90 COL  
90 90 COL  
90 90 COL  
90 90 COL  
90 90 COL  
90 90 COL  
90 90 COL  
90 90 COL  
40 90 VIGA  
40 90 VIGA  
40 90 VIGA  
40 90 VIGA  
40 90 VIGA  
40 90 VIGA  
40 90 VIGA  
40 90 VIGA  
40 90 VIGA  
40 90 VIGA  
40 90 VIGA  
-----  
(69, 69),(69, 69),(69, 69),(69, 69),(69, 69),(69, 69),(69, 69),  
(69, 69),(69, 69),(69, 69),(69, 69),(69, 69),(69, 69),(69, 69),  
(69, 69),(69, 69),(31, 69),(31, 69),(31, 69),(31, 69),(31, 69),  
(31, 69),(31, 69),(31, 69),(31, 69),(31, 69),(31, 69),(31, 69)
```

Figura 41: Primeras dimensiones sugeridas, β -beam.

Cabe señalar que este es un proceso que normalmente el programa realiza fuera de la vista del usuario. β -beam procede a reemplazar las secciones antiguas por las propuestas de manera automática y se vuelven a realizar todos los pasos anteriores, desde que se ejecutó el programa, dado que las secciones, en este punto, son actualizadas en todos sus atributos geométricos.

Tras 3 iteraciones, el factor de corrección tiende a 1 y para aproximaciones al centímetro se llega al siguiente resultado:

```
drift maximo: 1.634
ψ: 0.995063567597028
-----
verifica para el drift!
-----
64 64 COL
64 64 COL
64 64 COL
64 64 COL
64 64 COL
64 64 COL
64 64 COL
64 64 COL
64 64 COL
64 64 COL
64 64 COL
64 64 COL
64 64 COL
64 64 COL
64 64 COL
64 64 COL
64 64 COL
29 64 VIGA
29 64 VIGA
29 64 VIGA
29 64 VIGA
29 64 VIGA
29 64 VIGA
29 64 VIGA
29 64 VIGA
29 64 VIGA
29 64 VIGA
29 64 VIGA
29 64 VIGA
29 64 VIGA
29 64 VIGA
29 64 VIGA
-----
```

Figura 42: Verificación drift, tercera vuelta.

Habiendo cumplido, dentro de un margen de tolerancia dado por motivos constructivos donde se aproxima al centímetro, se procede al diseño de vigas y columnas propiamente tal.

B.4. Diseño de vigas y columnas

Se reciben los datos de los esfuerzos internos y se resume para 3 iteraciones, donde la primera no verifica esta condición, y las dos siguientes sí, pero con distintos costos.

A continuación, se presentan tablas resumen de estos resultados:

Tabla B.1: Medidas de las secciones en la Iteración 1.

Medidas en [cm]:

columnas							
ancho	largo						
60	60	60	60	60	60	60	60
55	55	55	55	55	55	55	55
50	50	50	50	50	50	50	50
45	45	45	45	45	45	45	45

vigas					
ancho	largo				
30	65	30	65	30	65
30	75	30	75	30	75
30	70	30	70	30	70
30	55	30	55	30	55

Tabla B.2: Costos y drift de la iteración 1.

valor columnas \$4.610.279
 valor acero \$3.615.370
 valor total \$8.225.649
 drift maximo anterior: 1.634

Tabla B.3: Medidas de las secciones en la iteración 2.

Medidas en [cm]:

columnas							
ancho	largo						
60	60	60	60	60	60	60	60
55	55	55	55	55	55	55	55
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

vigas					
ancho	largo				
30	75	30	75	30	75
40	80	40	80	40	80
30	65	30	65	30	65
30	65	30	65	30	65

Tabla B.4: Costos y drift de la iteración 2.

valor columnas \$4.360.465

valor acero \$3.665.044

valor total \$8.025.512

drift maximo : 1.643

Tabla B.5: Medidas de las secciones en la iteración 2.

Medidas en [cm]:

columnas							
ancho	largo						
55	55	55	55	55	55	55	55
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
45	45	45	45	45	45	45	45

vigas					
ancho	largo				
35	80	35	80	35	80
35	85	35	85	35	85
30	65	30	65	30	65
25	55	25	55	25	55

Tabla B.6: Costos y drift de la iteración 3.

valor columnas \$4.523.2332

valor acero \$3.704.413

valor total \$8.227.646

drift maximo : 1.631

B.5. Detallamiento final

En este ítem se recolecta toda la información relevante para el detallamiento de los elementos estructurales calculados, como lo son sus diámetros de barras de acero, sus longitudes y posiciones, las secciones de hormigón, entre otras, obtenidas por β -beam:

```
Columna n° 1, Tipo 1
Piso N° 1

Dimensiones
Largo : 3 [m]
Alto : 55 [cm]
Ancho : 55 [cm]

Refuerzo longitudinal
5 barras Ø 18 [mm] en la posición y = 5 [cm], área = 12.725 [cm2]
2 barras Ø 18 [mm] en la posición y = 16 [cm], área = 5.09 [cm2]
2 barras Ø 18 [mm] en la posición y = 27 [cm], área = 5.09 [cm2]
2 barras Ø 18 [mm] en la posición y = 38 [cm], área = 5.09 [cm2]
5 barras Ø 18 [mm] en la posición y = 50 [cm], área = 12.725 [cm2]

Cuantía = 0.01346

Columna para zonas laterales
Para unión superior

Longitud de empalme unión viga-columna = 35.9 [cm]
ngitud de gancho-remate = [182.6, 138.2] [cm]

Refuerzo transversal

Zonas de rótula plástica y nodo
Ubicación RP: de 0 - 55 [cm] y 165 - 220 [cm]:
Ubicación RP: de 220 - 300 [cm]:
N° ramas : 3
N° de estribos rótulas : 7
Espaciamiento : 8 [cm]
N° Estribos nodo : 12

Refuerzo exterior
Diámetro estribo exterior: 10 [cm]
Largo del estribo exterior 200.21 [cm]
Ubicación entre ejes de barras horizontales: x = 5 [cm] y x = 50 [cm]
Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 50 [cm]

Refuerzo interior
Diámetro de estribos y trabas interiores 10 [cm]
Largo de traba interior n°1 = 78.78 [cm]
Ubicación entre ejes de barras horizontales: x = 27 [cm]
Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 50 [cm]

Empalme central
Ubicación : de 55 - 165 [cm]
N° ramas : 3
N° de estribos : 13
Espaciamiento : 8 [cm]

Refuerzo exterior
Diámetro estribo exterior: 10 [cm]
Largo del estribo exterior 200.21 [cm]
Ubicación entre ejes de barras horizontales: x = 5 [cm] y x = 50 [cm]
Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 50 [cm]

Refuerzo interior
Diámetro de estribos y trabas interiores 10 [cm]
Largo de traba interior n°1 = 78.78 [cm]
Ubicación entre ejes de barras horizontales: x = 27 [cm]
Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 50 [cm]
Peso total de estribos : 57.3 [kg]
Peso total barras longitudinales : 18.5 [kg]
Volumen Hormigón : 0.9075000000000001 [m3]
Costo acero : $ 170800.0
Costo hormigón : $ 68062.5
Costo total de columna tipo: $ 238862.5
```

Resultados

Flexión: Mayor excentricidad 8.1 cm

Momentos

Mu_max = 47.13 [tf-m]

Momento nominal ajustado a Mu y Pu máximos

ØMn1 = 55.2 [tf-m]

Momento nominal ajustado a Mu máximo debido a mayor excentricidad:

ØMn2 = 30.6 [tf-m]

Cargas

Pu_max = 105.5 [tf]

Pu_min = 40.9 [tf]

Carga nominal que verifica Pu_max:

ØPn1 = 123.4 [tf]

Carga nominal que verifica Pu_min:

ØPn2 = 376.1 [tf]

F.U. 1 = 85.5 %

F.U. 2 = 10.9 %

Corte

Corte en zona de rótula plástica

ØVn1 = 61.8 [tf]

F.U.1 = 62.3 %

Corte en zona de empalme

ØVn2 = 61.8 [tf]

F.U.2 = 62.3 %

Columna n° 2, Tipo 1

Piso N° 2

Dimensiones

Largo : 3 [m]

Alto : 50 [cm]

Ancho : 50 [cm]

Refuerzo longitudinal

5 barras Ø 16 [mm] en la posición y = 5 [cm], área = 10.055 [cm²]

2 barras Ø 16 [mm] en la posición y = 15 [cm], área = 4.022 [cm²]

2 barras Ø 16 [mm] en la posición y = 25 [cm], área = 4.022 [cm²]

2 barras Ø 16 [mm] en la posición y = 35 [cm], área = 4.022 [cm²]

5 barras Ø 16 [mm] en la posición y = 45 [cm], área = 10.055 [cm²]

Cuantía = 0.01287

Columna para zonas laterales

Para unión superior

Longitud de gancho-remate = [162.2, 122.8] [cm]

Para unión inferior

Longitud de empalme unión viga-columna = 31.9 [cm]

fuerzo transversal

Zonas de rótula plástica y nodo

Ubicación RP: de 0 - 50 [cm] y 165 - 215 [cm]:

Ubicación RP: de 215 - 300 [cm]:

N° ramas : 3

N° de estribos rótulas : 7

Espaciamiento : 8 [cm]

° Estribos nodo : 13

Refuerzo exterior

Diámetro estribo exterior: 10 [cm]

Largo del estribo exterior 180.21 [cm]

Ubicación entre ejes de barras horizontales: x = 5 [cm] y x = 45 [cm]

Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 45 [cm]

Refuerzo interior

Diámetro de estribos y trabas interiores 10 [cm]

Largo de traba interior n°1 = 73.78 [cm]

Ubicación entre ejes de barras horizontales: x = 25 [cm]

Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 45 [cm]

Empalme central

Ubicación : de 50 - 165 [cm]
 N° ramas : 3
 N° de estribos : 12
 Espaciamiento : 9 [cm]

Refuerzo exterior
 Diámetro estribo exterior: 10 [cm]
 argo del estribo exterior 180.21 [cm]
 Ubicación entre ejes de barras horizontales: x = 5 [cm] y x = 45 [cm]
 Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 45 [cm]

Refuerzo interior
 Diámetro de estribos y trabas interiores 10 [cm]

Largo de traba interior n°1 = 73.78 [cm]
 Ubicación entre ejes de barras horizontales: x = 25 [cm]
 Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 45 [cm]
 Peso total de estribos : 48.5 [kg]
 Peso total barras longitudinales : 13.1 [kg]
 Volumen Hormigón : 0.75 [m3]: osto acero : \$ 136600.0
 Costo hormigón : \$ 56250.0
 Costo total de columna tipo: \$ 192850.0

Resultados

Flexión

Mayor excentricidad 8.0 cm

omentos

Mu_max = 31.83 [tf-m]
 Momento nominal ajustado a Mu y Pu máximos
 ØMn1 = 40.0 [tf-m]
 Momento nominal ajustado a Mu máximo debido a mayor excentricidad:
 Mn2 = 23.9 [tf-m]

Cargas

u_max = 77.2 [tf]
 Pu_min = 18.7 [tf]
 Carga nominal que verifica Pu_max:
 Pn1 = 97.1 [tf]
 Carga nominal que verifica Pu_min:
 ØPn2 = 296.7 [tf]
 U. 1 = 79.6 %
 F.U. 2 = 6.3 %

Corte

Corte en zona de rótula plástica
 ØVn1 = 55.6 [tf]
 F.U.1 = 59.5 %

Corte en zona de empalme
 ØVn2 = 49.5 [tf]
 F.U.2 = 66.9 %

Columna n° 3, Tipo 2
 Piso N° 1

Dimensiones
 Largo : 3 [m]
 Alto : 50 [cm]
 Ancho : 50 [cm]

Refuerzo longitudinal
 4 barras Ø 18 [mm] en la posición y = 5 [cm], área = 10.18 [cm²]
 2 barras Ø 18 [mm] en la posición y = 18 [cm], área = 5.09 [cm²]
 2 barras Ø 18 [mm] en la posición y = 32 [cm], área = 5.09 [cm²]
 4 barras Ø 18 [mm] en la posición y = 45 [cm], área = 10.18 [cm²]

Cuantía = 0.01222

Para unión superior
 Longitud de empalme unión viga-columna = 35.9 [cm]

efuerzo transversal

Zonas de rótula plástica y nodo

Ubicación RP: de 0 - 50 [cm] y 180 - 230 [cm]:
 Ubicación RP: de 230 - 300 [cm]:
 ° ramas : 4
 N° de estribos rótulas : 6
 Espaciamiento : 9 [cm]
 N° Estribos nodo : 12

Refuerzo exterior
 Diámetro estribo exterior: 10 [cm]
 Largo del estribo exterior 180.21 [cm]
 Ubicación entre ejes de barras horizontales: x = 5 [cm] y x = 45 [cm]
 Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 45 [cm]

Refuerzo interior
 Diámetro de estribos interiores 10 [cm]
 Largo de estribo interior n° 1 = 128.21 [cm]
 Ubicación entre ejes de barras horizontales: x = 18 [cm] y x = 32 [cm]
 Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 45 [cm]

Empalme central
 Ubicación : de 50 - 180 [cm]
 N° ramas : 4
 N° de estribos : 16
 Espaciamiento : 8 [cm]

Refuerzo exterior
 Diámetro estribo exterior: 10 [cm]
 Largo del estribo exterior 180.21 [cm]
 Ubicación entre ejes de barras horizontales: x = 5 [cm] y x = 45 [cm]
 Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 45 [cm]

Refuerzo interior
 Diámetro de estribos interiores 10 [cm]
 Largo de estribo interior n° 1 = 128.21 [cm]
 Ubicación entre ejes de barras horizontales: x = 18 [cm]
 Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 45 [cm]
 Peso total de estribos : 86.1 [kg]
 Peso total barras longitudinales : 13.9 [kg]
 Volumen Hormigón : 0.75 [m3]
 Costo acero : \$ 171200.0
 Costo hormigón : \$ 56250.0
 Costo total de columna tipo: \$ 227450.0

Resultados

Flexión: Mayor excentricidad 31.5 cm

Momentos
 Mu_max = 27.05 [tf-m]
 Momento nominal ajustado a Mu y Pu máximos
 ØMn1 = 34.1 [tf-m]
 Momento nominal ajustado a Mu máximo debido a mayor excentricidad:
 ØMn2 = 39.6 [tf-m]

Cargas
 Pu_max = 46.29 [tf]
 Pu_min = 12.4 [tf]
 Carga nominal que verifica Pu_max:
 ØPn1 = 58.5 [tf]
 Carga nominal que verifica Pu_min:
 ØPn2 = 125.6 [tf]
 F.U. 1 = 79.3 %
 F.U. 2 = 9.9 %

Corte
 Corte en zona de rótula plástica
 ØVn1 = 65.9 [tf]
 F.U.1 = 70.6 %
 Corte en zona de empalme
 ØVn2 = 74.2 [tf]
 F.U.2 = 37.7 %

Columna n° 4, Tipo 2
 Piso N° 2

Dimensiones
 Largo : 3 [m]
 Alto : 45 [cm]
 Ancho : 45 [cm]

Refuerzo longitudinal
 4 barras Ø 28 [mm] en la posición y = 5 [cm], área = 24.632 [cm2]
 2 barras Ø 28 [mm] en la posición y = 17 [cm], área = 12.316 [cm2]

2 barras Ø 28 [mm] en la posición y = 28 [cm], área = 12.316 [cm²]
4 barras Ø 28 [mm] en la posición y = 40 [cm], área = 24.632 [[cm²]

Cuantía = 0.03649

Para unión superior
Longitud de gancho-remate = [238.0, 169] [cm]
Para unión inferior

ngitud de gancho-remate = [238.0, 169] [cm]

Refuerzo transversal

onas de rótula plástica y nodo
Ubicación RP: de 0 - 50.0 [cm] y 180.0 - 230 [cm]:
Ubicación RP: de 230 - 300 [cm]:
N° ramas : 4
N° de estribos rótulas : 6
Espaciamiento : 9 [cm]
N° Estribos nodo : 12

Refuerzo exterior

Diámetro estribo exterior: 10 [cm]
Largo del estribo exterior 160.21 [cm]
Ubicación entre ejes de barras horizontales: x = 5 [cm] y x = 40 [cm]
Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 40 [cm]

Refuerzo interior

Diámetro de estribos interiores 10 [cm]
Largo de estribo interior n° 1 = 112.21 [cm]
Ubicación entre ejes de barras horizontales: x = 17 [cm] y x = 28 [cm]
Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 40 [cm]

Empalme central

Ubicación : de 50.0 - 180.0 [cm]
N° ramas : 4
N° de estribos : 14
Espaciamiento : 9 [cm]

Refuerzo exterior

Diámetro estribo exterior: 10 [cm]
Largo del estribo exterior 160.21 [cm]
Ubicación entre ejes de barras horizontales: x = 5 [cm] y x = 40 [cm]
Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 40 [cm]

Refuerzo interior

Diámetro de estribos interiores 10 [cm]
Largo de estribo interior n° 1 = 112.21 [cm]
Ubicación entre ejes de barras horizontales: x = 17 [cm]
Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 40 [cm]
Peso total de estribos : 66.4 [kg]
Peso total barras longitudinales : 51.6 [kg]
Volumen Hormigón : 0.6075 [m³]
Costo acero : \$ 290300.0
Costo hormigón : \$ 45562.5
Costo total de columna tipo: \$ 335862.5

Resultados

Flexión: Mayor excentricidad 27.9 cm

Momentos

Mu_max = 28.06 [tf-m]
Momento nominal ajustado a Mu y Pu máximos
ØMn1 = 50.9 [tf-m]
Momento nominal ajustado a Mu máximo debido a mayor excentricidad:
ØMn2 = 39.7 [tf-m]

Cargas

Pu_max = 21.95 [tf]
Pu_min = 8.6 [tf]
Carga nominal que verifica Pu_max:
ØPn1 = 39.8 [tf]
Carga nominal que verifica Pu_min:
ØPn2 = 142.5 [tf]
F.U. 1 = 55.1 %
F.U. 2 = 6.0 %

Corte

Corte en zona de rótula plástica
Øvn1 = 58.6 [tf]
F.U.1 = 61.8 %
Corte en zona de empalme
Øvn2 = 58.6 [tf]
F.U.2 = 43.7 %

Columna n° 5, Tipo 2
Piso N° 1

Dimensiones

Largo : 3 [m]
Alto : 65 [cm]
Ancho : 65 [cm]

Refuerzo longitudinal

6 barras Ø 18 [mm] en la posición y = 5 [cm], área = 15.27 [cm²]
2 barras Ø 18 [mm] en la posición y = 16 [cm], área = 5.09 [cm²]
2 barras Ø 18 [mm] en la posición y = 27 [cm], área = 5.09 [cm²]
2 barras Ø 18 [mm] en la posición y = 38 [cm], área = 5.09 [cm²]
2 barras Ø 18 [mm] en la posición y = 49 [cm], área = 5.09 [cm²]
6 barras Ø 18 [mm] en la posición y = 60 [cm], área = 15.27 [cm²]

Cuantía = 0.01205
remates

Columna para zonas centrales
Para unión superior

Longitud de empalme unión viga-columna = 35.9 [cm]

Refuerzo transversal

onas de rótula plástica y nodo

Ubicación RP: de 0 - 65 [cm] y 155 - 220 [cm]:

Ubicación RP: de 220 - 300 [cm]:

N° ramas : 4

N° de estribos rótulas : 9

Espaciamiento : 8 [cm]

N° Estribos nodo : 9

Refuerzo exterior

Diámetro estribo exterior: 12 [cm]

Largo del estribo exterior 245.25 [cm]

Ubicación entre ejes de barras horizontales: x = 5 [cm] y x = 60 [cm]

Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 60 [cm]

Refuerzo interior

Diámetro de estribos interiores 10 [cm]

Largo de estribo interior n° 1 = 152.21 [cm]

Ubicación entre ejes de barras horizontales: x = 27 [cm] y x = 38 [cm]

Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 60 [cm]

Empalme central

Ubicación : de 65 - 155 [cm]

N° ramas : 4

N° de estribos : 11

Espaciamiento : 8 [cm]

Refuerzo exterior

Diámetro estribo exterior: 10 [cm]

Largo del estribo exterior 240.21 [cm]

Ubicación entre ejes de barras horizontales: x = 5 [cm] y x = 60 [cm]

Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 60 [cm]

Refuerzo interior

Diámetro de estribos interiores 10 [cm]

Largo de estribo interior n° 1 = 152.21 [cm]

Ubicación entre ejes de barras horizontales: x = 27 [cm]

Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 60 [cm]

Peso total de estribos : 81.5 [kg]

Peso total barras longitudinales : 23.2 [kg]

Volumen Hormigón : 1.2675 [m³]

Costo acero : \$ 223300.0

Costo hormigón : \$ 95062.5

Costo total de columna tipo: \$ 318362.5

Resultados

Flexión: Mayor excentricidad 113.3 cm

Momentos

Mu_max = 49.56 [tf-m]

Momento nominal ajustado a Mu y Pu máximos

ØMn1 = 77.7 [tf-m]

Momento nominal ajustado a Mu máximo debido a mayor excentricidad:

ØMn2 = 67.8 [tf-m]

Cargas

Pu_max = 168.08 [tf]

Pu_min = 48.7 [tf]

Carga nominal que verifica Pu_max:
ØPn1 = 263.3 [tf]
Carga nominal que verifica Pu_min:
ØPn2 = 59.8 [tf]
F.U. 1 = 63.8 %
F.U. 2 = 81.4 %

Corte
Corte en zona de rótula plástica
Øvn1 = 120.7 [tf]
F.U.1 = 89.4 %
Corte en zona de empalme
Øvn2 = 98.9 [tf]
F.U.2 = 61.3 %

Columna n° 6, Tipo 2
Piso N° 2

Dimensiones
Largo : 3 [m]
Alto : 60 [cm]
Ancho : 60 [cm]

Refuerzo longitudinal
5 barras Ø 18 [mm] en la posición y = 5 [cm], área = 12.725 [cm2]
2 barras Ø 18 [mm] en la posición y = 18 [cm], área = 5.09 [cm2]
2 barras Ø 18 [mm] en la posición y = 30 [cm], área = 5.09 [cm2]
2 barras Ø 18 [mm] en la posición y = 42 [cm], área = 5.09 [cm2]
5 barras Ø 18 [mm] en la posición y = 55 [cm], área = 12.725 [cm2]

Cuantía = 0.01131

Columna para zonas centrales
Para unión superior

Longitud de gancho-remate = [182.6, 138.2] [cm]
Para unión inferior
ongitud de gancho-remate = [182.6, 138.2] [cm]

Refuerzo transversal
Zonas de rótula plástica y nodo
Ubicación RP: de 0 - 60 [cm] y 155 - 215 [cm]:
Ubicación RP: de 215 - 300 [cm]:
N° ramas : 4
N° de estribos rótulas : 7
Espaciamiento : 9 [cm]
N° Estribos nodo : 13

Refuerzo exterior
Diámetro estribo exterior: 10 [cm]
Largo del estribo exterior 220.21 [cm]
Ubicación entre ejes de barras horizontales: x = 5 [cm] y x = 55 [cm]
Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 55 [cm]

Refuerzo interior
Diámetro de estribos interiores 10 [cm]
Largo de estribo interior n° 1 = 168.21 [cm]
Ubicación entre ejes de barras horizontales: x = 18 [cm] y x = 42 [cm]
Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 55 [cm]

Empalme central
Ubicación : de 60 - 155 [cm]
N° ramas : 3
N° de estribos : 11
Espaciamiento : 8 [cm]

Refuerzo exterior
Diámetro estribo exterior: 10 [cm]
Largo del estribo exterior 220.21 [cm]
Ubicación entre ejes de barras horizontales: x = 5 [cm] y x = 55 [cm]
Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 55 [cm]

Refuerzo interior
Diámetro de estribos y trabas interiores 10 [cm]
Largo de traba interior n°1 = 83.78 [cm]
Ubicación entre ejes de barras horizontales: x = 18 [cm]
Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 55 [cm]
Peso total de estribos : 64.0 [kg]
Peso total barras longitudinales : 18.5 [kg]
Volumen Hormigón : 1.08 [m3]
Costo acero : \$ 177500.0
Costo hormigón : \$ 81000.0
Costo total de columna tipo: \$ 258500.0

Resultados

Flexión: Mayor excentricidad 103.4 cm

Momentos

Mu_max = 40.38 [tf-m]

Momento nominal ajustado a Mu y Pu máximos

ØMn1 = 65.0 [tf-m]

Momento nominal ajustado a Mu máximo debido a mayor excentricidad:

ØMn2 = 50.5 [tf-m]

Cargas

Pu_max = 125.89 [tf]

Pu_min = 37.8 [tf]

Carga nominal que verifica Pu_max:

ØPn1 = 202.2 [tf]

Carga nominal que verifica Pu_min:

ØPn2 = 48.8 [tf]

F.U. 1 = 62.3 %

F.U. 2 = 77.4 %

Corte

Corte en zona de rótula plástica

Øvn1 = 80.6 [tf]

F.U.1 = 99.2 %

Corte en zona de empalme

Øvn2 = 68.0 [tf]

F.U.2 = 61.6 %

Columna n° 7, Tipo 2

Piso N° 1

Dimensiones

Largo : 3 [m]

Alto : 55 [cm]

Ancho : 55 [cm]

Refuerzo longitudinal

5 barras Ø 16 [mm] en la posición y = 5 [cm], área = 10.055 [cm²]

2 barras Ø 16 [mm] en la posición y = 16 [cm], área = 4.022 [cm²]

2 barras Ø 16 [mm] en la posición y = 27 [cm], área = 4.022 [cm²]

2 barras Ø 16 [mm] en la posición y = 38 [cm], área = 4.022 [cm²]

5 barras Ø 16 [mm] en la posición y = 50 [cm], área = 10.055 [[cm]²]

Cuantía = 0.01064

Columna para zonas centrales

Para unión superior

Longitud de empalme unión viga-columna = 31.9 [cm]

Refuerzo transversal

nas de rótula plástica y nodo

Ubicación RP: de 0 - 55 [cm] y 175 - 230 [cm]:

Ubicación RP: de 230 - 300 [cm]:

N° ramas : 3

N° de estribos rótulas : 7

Espaciamiento : 8 [cm]

N° Estribos nodo : 11

Refuerzo exterior

Diámetro estribo exterior: 10 [cm]

Largo del estribo exterior 200.21 [cm]

Ubicación entre ejes de barras horizontales: x = 5 [cm] y x = 50 [cm]

Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 50 [cm]

Refuerzo interior

Diámetro de estribos y trabas interiores 10 [cm]

Largo de traba interior n°1 = 78.78 [cm]

Ubicación entre ejes de barras horizontales: x = 27 [cm]

Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 50 [cm]

Empalme central

Ubicación : de 55 - 175 [cm]

N° ramas : 3

N° de estribos : 13

Espaciamiento : 9 [cm]

Refuerzo exterior

Diámetro estribo exterior: 10 [cm]

Largo del estribo exterior 200.21 [cm]

Ubicación entre ejes de barras horizontales: x = 5 [cm] y x = 50 [cm]

Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 50 [cm]

Refuerzo interior
 Diámetro de estribos y trabas interiores 10 [cm]
 Largo de traba interior n°1 = 78.78 [cm]
 Ubicación entre ejes de barras horizontales: x = 27 [cm]
 Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 50 [cm]
 Peso total de estribos : 57.3 [kg]
 Peso total barras longitudinales : 13.1 [kg]
 Volumen Hormigón : 0.9075000000000001 [m3]
 Costo acero : \$ 145500.0
 Costo hormigón : \$ 68062.5
 Costo total de columna tipo: \$ 213562.5

Resultados

Flexión: Mayor excentricidad 88.5 cm

Momentos
 Mu_max = 31.74 [tf-m]
 Momento nominal ajustado a Mu y Pu máximos
 ØMn1 = 50.1 [tf-m]
 Momento nominal ajustado a Mu máximo debido a mayor excentricidad:
 ØMn2 = 36.7 [tf-m]

Cargas
 Pu_max = 81.99 [tf]
 Pu_min = 30.4 [tf]
 Carga nominal que verifica Pu_max:
 ØPn1 = 129.4 [tf]
 Carga nominal que verifica Pu_min:
 ØPn2 = 41.5 [tf]
 F.U. 1 = 63.4 %
 F.U. 2 = 73.3 %

Corte
 Corte en zona de rótula plástica
 ØVn1 = 61.8 [tf]
 F.U.1 = 87.5 %
 Corte en zona de empalme
 ØVn2 = 55.0 [tf]
 F.U.2 = 59.1 %

Columna n° 8, Tipo 2
 Piso N° 2

Dimensiones
 Largo : 3 [m]
 Alto : 50 [cm]
 Ancho : 50 [cm]

Refuerzo longitudinal
 4 barras Ø 18 [mm] en la posición y = 5 [cm], área = 10.18 [cm2]
 2 barras Ø 18 [mm] en la posición y = 18 [cm], área = 5.09 [cm2]
 2 barras Ø 18 [mm] en la posición y = 32 [cm], área = 5.09 [cm2]
 4 barras Ø 18 [mm] en la posición y = 45 [cm], área = 10.18 [[cm]2]

Cuantía = 0.01222

Para unión superior
 Longitud de gancho-remate = [182.6, 138.2] [cm]
 Para unión inferior

ngitud de gancho-remate = [182.6, 138.2] [cm]

Refuerzo transversal

onas de rótula plástica y nodo
 Ubicación RP: de 0 - 50 [cm] y 180 - 230 [cm]:
 Ubicación RP: de 230 - 300 [cm]:
 N° ramas : 4
 N° de estribos rótulas : 6
 Espaciamiento : 9 [cm]
 N° Estribos nodo : 12

Refuerzo exterior
 Diámetro estribo exterior: 10 [cm]
 Largo del estribo exterior 180.21 [cm]
 Ubicación entre ejes de barras horizontales: x = 5 [cm] y x = 45 [cm]
 Ubicación entre ejes de barras verticales: y = 5 [cm] e y = 45 [cm]

Refuerzo interior
 Diámetro de estribos interiores 10 [cm]
 Largo de estribo interior n° 1 = 128.21 [cm]
 Ubicación entre ejes de barras horizontales: x = 18 [cm] y x = 32 [cm]

Ubicación entre ejes de barras verticales: $y = 5$ [cm] e $y = 45$ [cm]

Empalme central
 Ubicación : de 50 - 180 [cm]
 N° ramas : 4
 N° de estribos : 16
 Espaciamiento : 8 [cm]

Refuerzo exterior
 Diámetro estribo exterior: 10 [cm]
 Largo del estribo exterior 180.21 [cm]
 Ubicación entre ejes de barras horizontales: $x = 5$ [cm] y $x = 45$ [cm]
 Ubicación entre ejes de barras verticales: $y = 5$ [cm] e $y = 45$ [cm]

Refuerzo interior
 Diámetro de estribos interiores 10 [cm]
 Largo de estribo interior n° 1 = 128.21 [cm]
 Ubicación entre ejes de barras horizontales: $x = 18$ [cm]
 Ubicación entre ejes de barras verticales: $y = 5$ [cm] e $y = 45$ [cm]
 Peso total de estribos : 86.1 [kg]
 Peso total barras longitudinales : 13.9 [kg]
 Volumen Hormigón : 0.75 [m3]
 Costo acero : \$ 171200.0
 Costo hormigón : \$ 56250.0
 Costo total de columna tipo: \$ 227450.0

Resultados

Flexión: Mayor excentricidad 52.6 cm

Momentos
 $Mu_{max} = 28.06$ [tf-m]
 Momento nominal ajustado a Mu y Pu máximos
 $\phi Mn1 = 32.7$ [tf-m]
 Momento nominal ajustado a Mu máximo debido a mayor excentricidad:
 $\phi Mn2 = 35.2$ [tf-m]

Cargas
 $Pu_{max} = 41.41$ [tf]
 $Pu_{min} = 24.9$ [tf]
 Carga nominal que verifica Pu_{max} :
 $\phi Pn1 = 48.1$ [tf]
 Carga nominal que verifica Pu_{min} :
 $\phi Pn2 = 67.0$ [tf]
 $F.U. 1 = 86.1$ %
 $F.U. 2 = 37.2$ %

Corte
 Corte en zona de rótula plástica
 $\phi Vn1 = 65.9$ [tf]
 $F.U.1 = 56.2$ %
 Corte en zona de empalme
 $\phi Vn2 = 74.2$ [tf]
 $F.U.2 = 35.2$ %
 Reporte de diseño y ubicación de la estructura

cantidad pisos 4
 cantidad vigas tipo por piso 1

Viga n° 1
 Viga tipo del piso 1

Dimensiones

Largo : 6.3 [m]
 Alto : 80 [cm]
 Ancho : 35 [cm]

Refuerzo longitudinal

Armadura superior principal
 2 barras $\phi 22$ [cm] , 1 barra $\phi 18$ [mm] en la posición $y = 5$ [cm], área = 10.15 [cm2]
 Armadura suplementaria
 2 barras $\phi 18$ [cm] , 1 barra $\phi 22$ [mm] en la posición $y = 10.5$ [cm], área = 8.89 [cm2]
 Armadura lateral
 2 barras $\phi 12$ [mm] en la posición $y = 32.0$ cm, área = 2.26 [cm2]
 2 barras $\phi 12$ [mm] en la posición $y = 54.0$ cm, área = 2.26 [cm2]
 Armadura inferior principal
 2 barras $\phi 25$ [cm]

Cuantías
 Superior = 0.0073
 Inferior = 0.0037

Cubicación de acero en barras longitudinales.
 Largos de suples :
 0.25lo : 157.5 [cm]
 0.3lo : 189.0 [cm]
 kg de barras de suples : 65.5 [kg]
 kg de otras barras : 384.8 [kg]
 Volumen de hormigón : 5.292 [m3]
 Costo de acero: \$ 450200.0
 Costo de hormigón : \$ 396900.0

Barras superiores :
 Longitud de traslapo de armadura superior : 141.3 [cm]
 Desarrollo de gancho : 34.4 [cm]
 Gancho : 52.5 [cm]
 Diámetro : 22 [mm]
 12db : 26.4 [cm]

Barras suplementarias :
 Desarrollo de gancho : 34.4 [cm]
 Gancho : 52.5 [cm]
 Diámetro : 22 [mm]
 12db : 26.4 [cm]

Barras laterales :
 Longitud de traslapo de armadura lateral : 48.0 [cm]
 Desarrollo de gancho : 18.8 [cm]
 Gancho : 26.4 [cm]
 Diámetro : 12.0 [mm]
 12db : 14.399999999999999 [cm]

Barras inferiores :
 Longitud de traslapo de armadura inferior : 123.5 [cm]
 Desarrollo de gancho : 39.1 [cm]
 Gancho : 60.3 [cm]
 Diámetro : 25 [mm]
 12db : 30.0 [cm]

Refuerzo transversal
 Zonas de rótula plástica, de 0 a 160 [cm] y 470.0 a 630.0 [cm]:
 Diámetro : 10 [cm]
 N° ramas : 2
 Estribos = 1
 Largo de estribo n° 1 = 200.21 [cm]
 Traba central: no
 Espaciamiento : 10 [cm]
 N° estribos : 16 en cada extremo
 Volumen de acero en zona de rótulas plásticas : 5030.4 [cm3]
 Peso del acero 39.5 [kg]

Zonas central, de 160 a 470.0 [cm]:
 Diámetro : 10 [cm]
 N° ramas : 2
 longitud de empalme barras inferiores : 141.3 [cm]
 longitud de empalme barras superiores : 123.5 [cm]
 espaciamiento normal : 17 [cm]
 espaciamiento zona de empalme : 10 [cm]
 Largo de estribo n° 1 = 200.21 [cm]
 Traba central: no
 Espaciamiento : 17 [cm]
 N° estribos : 29
 Volumen de acero en zona central : 4558.799999999999 [cm3]
 Peso del acero 35.8 [kg]

Trabas Horizontales
 N° Trabas por estribo = 3
 N° de estribos donde va traba = 61
 Largo trabas = 58.78 [cm]
 Volumen de acero en trabas horizontales : 12155.1 [cm3]
 Peso del acero : 95.4 [kg]

Acero total : 170.7 [kg]
 Hormigón total : 5.29 [m3]
 Costo hormigón : \$ 396900.0
 Costo acero : \$ 620900.0
 Costo total de vigas por piso : \$ 1017800.0
 Resultados del análisis

Flexión
 $\emptyset M_{n+}$ = 54.3 [tf-m]
 $\emptyset M_{n-}$ = -34.0 [tf-m]
 F.U. mayor = 92.4 %

Corte
 $\emptyset V_{n1}$ = 49.5 [tf]
 F.U.1 = 87.7 %
 $\emptyset V_{n2}$ = 29.1 [tf]

F.U.2 = 99.7 %

Viga n° 2
Viga tipo del piso 2

Dimensiones

Largo : 6.35 [m]
Alto : 85 [cm]
Ancho : 35 [cm]

Refuerzo longitudinal

Armadura superior principal
2 barras Ø 22 [cm] , 1 barra Ø 22[mm] en la posición y = 5 [cm], área = 11.4 [cm2]
Armadura suplementaria
2 barras Ø 22 [cm] , 1 barra Ø 18[mm] en la posición y = 10.5 [cm], área = 10.15 [cm2]
Armadura lateral
2 barras Ø 12 [mm] en la posición y = 34.0 cm, área = 2.26 [cm2]
2 barras Ø 12 [mm] en la posición y = 57.0 cm, área = 2.26 [cm2]
Armadura inferior principal
2 barras Ø 28 [cm]

Cuantías
Superior = 0.0077
Inferior = 0.0044

Cubicación de acero en barras longitudinales.

Largos de suples :
0.25lo : 158.8 [cm]
0.3lo : 190.5 [cm]
kg de barras de suples : 74.4 [kg]
kg de otras barras : 453.5 [kg]
Volumen de hormigón : 5.667375 [m3]
Costo de acero: \$ 527900.0
Costo de hormigón : \$ 425053.125

Barras superiores :
Longitud de traslapo de armadura superior : 141.3 [cm]
Desarrollo de gancho : 34.4 [cm]
Gancho : 52.5 [cm]
Diámetro : 22 [mm]
12db : 26.4 [cm]

Barras suplementarias :
Desarrollo de gancho : 34.4 [cm]
Gancho : 52.5 [cm]
Diámetro : 22 [mm]
12db : 26.4 [cm]

Barras laterales :
Longitud de traslapo de armadura lateral : 48.0 [cm]
Desarrollo de gancho : 18.8 [cm]
Gancho : 26.4 [cm]
Diámetro : 12.0 [mm]
12db : 14.399999999999999 [cm]

Barras inferiores :
Longitud de traslapo de armadura inferior : 138.4 [cm]
Desarrollo de gancho : 43.8 [cm]
Gancho : 68.2 [cm]
Diámetro : 28 [mm]
12db : 33.6 [cm]

Refuerzo transversal
Zonas de rótula plástica, de 0 a 170 [cm] y 465.0 a 635.0 [cm]:
Diámetro : 10 [cm]
N° ramas : 3
Estribos = 1
Largo de estribo n° 1 = 210.21 [cm]
Traba central: si
Largo de traba = 108.78 [cm]
Espaciamiento : 13 [cm]
N° estribos : 13 en cada extremo
Volumen de acero en zona de rótulas plásticas : 6510.4 [cm3]
Peso del acero 51.1 [kg]

Zonas central, de 170 a 465.0 [cm]:
Diámetro : 10 [cm]
N° ramas : 2
longitud de empalme barras inferiores : 141.3 [cm]
longitud de empalme barras superiores : 138.4 [cm]
espaciamiento normal : 16 [cm]
espaciamiento zona de empalme : 10[cm]
Largo de estribo n° 1 = 210.21 [cm]

Traba central: no
 Espaciamiento : 16 [cm]
 N° estribos : 28
 Volumen de acero en zona central : 4620.0 [cm³]
 Peso del acero 55.0 [kg]

 Trabas Horizontales
 N° Trabas por estribo = 3
 N° de estribos donde va traba = 54
 Largo trabas = 58.78 [cm]
 Volumen de acero en trabas horizontales : 10760.3 [cm³]
 Peso del acero : 84.5 [kg]

 Acero total : 190.6 [kg]
 Hormigón total : 5.67 [m³]
 Costo hormigón : \$ 425053.125
 Costo acero : \$ 718500.0
 Costo total de vigas por piso : \$ 1143553.125
 Resultados del análisis

Flexión
 ØMn+ = 65.2 [tf-m]
 ØMn- = -43.2 [tf-m]
 F.U. mayor = 90.3 %

Corte
 ØVn1 = 60.9 [tf]
 F.U.1 = 80.5 %
 ØVn2 = 33.0 [tf]
 F.U.2 = 99.5 %

Viga n° 3, viga tipo del piso 3

Dimensiones

Largo : 6.4 [m]
 Alto : 65 [cm]
 Ancho : 30 [cm]

Refuerzo longitudinal

Armadura superior principal
 2 barras Ø 22 [cm] , 1 barra Ø 22 [mm] en la posición y = 5 [cm], área = 11.4 [cm²]
 Armadura suplementaria
 2 barras Ø 22 [cm] , 1 barra Ø 18 [mm] en la posición y = 10.5 [cm], área = 10.15 [cm²]
 Armadura lateral
 2 barras Ø 12 [mm] en la posición y = 10.5 cm, área = 2.26 [cm²]
 2 barras Ø 12 [mm] en la posición y = 35.0 cm, área = 2.26 [cm²]
 Armadura inferior principal
 2 barras Ø 18 [cm] , 1 barra Ø 22 [mm] en la posición y = 60 [cm], área = 8.89 [cm²]

Cuantías
 Superior = 0.012
 Inferior = 0.0049

Cubicación de acero en barras longitudinales.

Largos de suples :
 0.25lo : 160.0 [cm]
 0.3lo : 192.0 [cm]
 kg de barras de suples : 74.0 [kg]
 kg de otras barras : 390.1 [kg]
 Volumen de hormigón : 3.744 [m³]
 Costo de acero: \$ 464100.0
 Costo de hormigón : \$ 280800.0

Barras superiores :
 Longitud de traslapo de armadura superior : 141.3 [cm]
 Desarrollo de gancho : 34.4 [cm]
 Gancho : 52.5 [cm]
 Diámetro : 22 [mm]
 12db : 26.4 [cm]

Barras suplementarias :
 Desarrollo de gancho : 34.4 [cm]
 Gancho : 52.5 [cm]
 Diámetro : 22 [mm]
 12db : 26.4 [cm]

Barras laterales :
 Longitud de traslapo de armadura lateral : 48.0 [cm]
 Desarrollo de gancho : 18.8 [cm]
 Gancho : 26.4 [cm]
 Diámetro : 12.0 [mm]
 12db : 14.399999999999999 [cm]

Barras inferiores :

Longitud de traslapo de armadura inferior : 108.7 [cm]
Desarrollo de gancho : 34.4 [cm]
Gancho : 52.5 [cm]
Diámetro : 22 [mm]
12db : 26.4 [cm]

Refuerzo transversal
Zonas de rótula plástica, de 0 a 130 [cm] y 510.0 a 640.0 [cm]:
Diámetro : 10 [cm]
N° ramas : 2
Estribos = 1
Largo de estribo n° 1 = 170.21 [cm]
Traba central: no
Espaciamiento : 10 [cm]
N° estribos : 13 en cada extremo
Volumen de acero en zona de rótulas plásticas : 3473.6 [cm3]
Peso del acero 27.3 [kg]

Zonas central, de 130 a 510.0 [cm]:
Diámetro : 10 [cm]
N° ramas : 2
longitud de empalme barras inferiores : 141.3 [cm]
longitud de empalme barras superiores : 108.7 [cm]
espaciamiento normal : 13 [cm]
espaciamiento zona de empalme : 10 [cm]
Largo de estribo n° 1 = 170.21 [cm]
Traba central: no
Espaciamiento : 13 [cm]
N° estribos : 34
Volumen de acero en zona central : 4542.4 [cm3]
Peso del acero 35.7 [kg]

Trabas Horizontales
N° Trabas por estribo = 2
N° de estribos donde va traba = 60
Largo trabas = 53.78 [cm]
Volumen de acero en trabas horizontales : 7292.6 [cm3]
Peso del acero : 57.2 [kg]

Acero total : 120.2 [kg]
Hormigón total : 3.74 [m3]
Costo hormigón : \$ 280800.0
Costo acero : \$ 584300.0
Costo total de vigas por piso : \$ 865100.0
Resultados del análisis

Flexión
ØMn+ = 44.7 [tf-m]
ØMn- = -22.8 [tf-m]
F.U. mayor = 92.7 %

Corte
ØVn1 = 39.6 [tf]
F.U.1 = 97.5 %
ØVn2 = 30.4 [tf]
F.U.2 = 89.9 %

Viga n° 4
Viga tipo del piso 4

Dimensiones

Largo : 6.4 [m]
Alto : 55 [cm]
Ancho : 25 [cm]

Refuerzo longitudinal

Armadura superior principal
2 barras Ø 28 [cm]
Armadura suplementaria
2 barras Ø 28 [cm]
Armadura lateral
2 barras Ø 12 [mm] en la posición y = 12.0 cm, área = 2.26 [cm2]
2 barras Ø 12 [mm] en la posición y = 31.0 cm, área = 2.26 [cm2]
Armadura inferior principal
2 barras Ø 18 [cm]

Cuantías
Superior = 0.0197
Inferior = 0.0041

Cubicación de acero en barras longitudinales.
Largos de suples :
0.25lo : 160.0 [cm]
0.3lo : 192.0 [cm]

kg de barras de suples : 92.9 [kg]
 kg de otras barras : 343.7 [kg]
 Volumen de hormigón : 2.64 [m3]
 Costo de acero: \$ 436500.0
 Costo de hormigón : \$ 198000.0

Barras superiores :
 Longitud de traslapo de armadura superior : 179.9 [cm]
 Desarrollo de gancho : 43.8 [cm]
 Gancho : 68.2 [cm]
 Diámetro : 28 [mm]
 12db : 33.6 [cm]

Barras suplementarias :
 Desarrollo de gancho : 43.8 [cm]
 Gancho : 68.2 [cm]
 Diámetro : 28 [mm]
 12db : 33.6 [cm]

Barras laterales :
 Longitud de traslapo de armadura lateral : 48.0 [cm]
 Desarrollo de gancho : 18.8 [cm]
 Gancho : 26.4 [cm]
 Diámetro : 12.0 [mm]
 12db : 14.399999999999999 [cm]

Barras inferiores :
 Longitud de traslapo de armadura inferior : 72.0 [cm]
 Desarrollo de gancho : 28.1 [cm]
 Gancho : 42.0 [cm]
 Diámetro : 18 [mm]
 12db : 21.599999999999998 [cm]

Refuerzo transversal
 Zonas de rótula plástica, de 0 a 110 [cm] y 530.0 a 640.0 [cm]:
 Diámetro : 10 [cm]
 N° ramas : 2
 Estribos = 1
 Largo de estribo n° 1 = 150.21 [cm]
 Traba central: no
 Espaciamiento : 9 [cm]
 N° estribos : 12 en cada extremo
 Volumen de acero en zona de rótulas plásticas : 2829.6 [cm3]
 Peso del acero 22.2 [kg]

Zonas central, de 110 a 530.0 [cm]:
 Diámetro : 10 [cm]
 N° ramas : 2
 longitud de empalme barras inferiores : 179.9 [cm]
 longitud de empalme barras superiores : 72.0 [cm]
 espaciamiento normal : 13 [cm]
 espaciamiento zona de empalme : 10 [cm]
 Largo de estribo n° 1 = 150.21 [cm]
 Traba central: no
 Espaciamiento : 13 [cm]
 N° estribos : 38
 Volumen de acero en zona central : 4480.2 [cm3]
 Peso del acero 35.2 [kg]

Trabas Horizontales
 N° Trabas por estribo = 2
 N° de estribos donde va traba = 62
 Largo trabas = 48.78 [cm]
 Volumen de acero en trabas horizontales : 6835.1 [cm3]
 Peso del acero : 53.7 [kg]

Acero total : 111.1 [kg]
 Hormigón total : 2.64 [m3]
 Costo hormigón : \$ 198000.0
 Costo acero : \$ 547600.0
 Costo total de vigas por piso : \$ 745600.0
 Resultados del análisis

Flexión
 ØMn+ = 36.5 [tf-m]
 ØMn- = -13.7 [tf-m]
 F.U. mayor = 88.7 %

Corte
 ØVn1 = 36.6 [tf]
 F.U.1 = 95.6 %
 ØVn2 = 25.4 [tf]
 F.U.2 = 99.2 %

B.6. Curvas de interacción de columnas tipo por piso

En este ítem se muestran los gráficos obtenidos a partir del cálculo anterior.

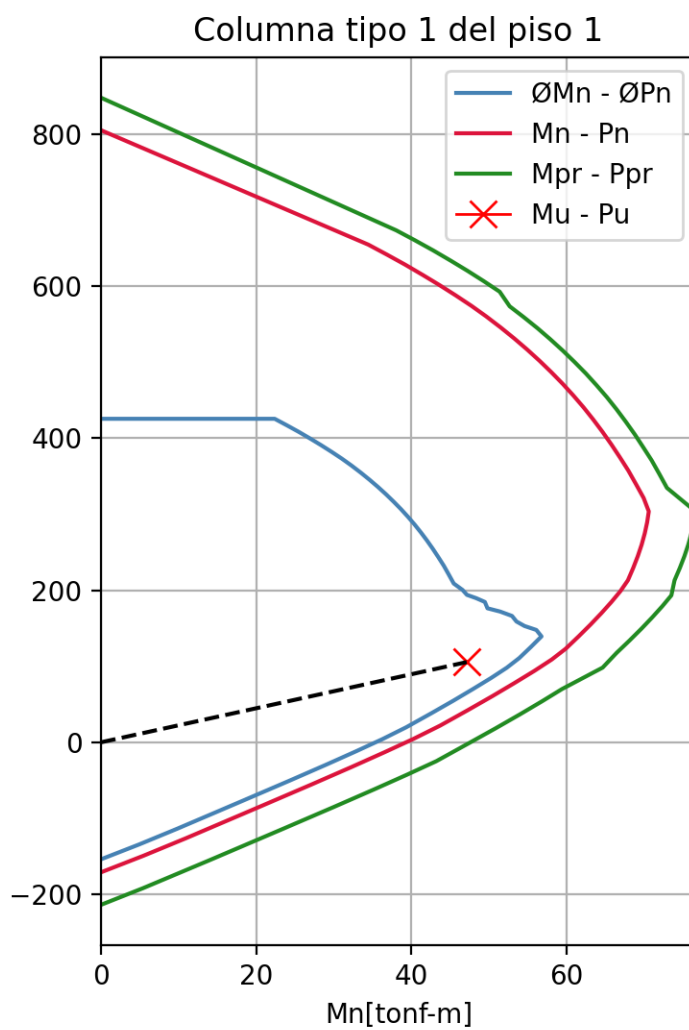


Figura 43: Curva de interacción de la columna del tipo 1 del piso 1.

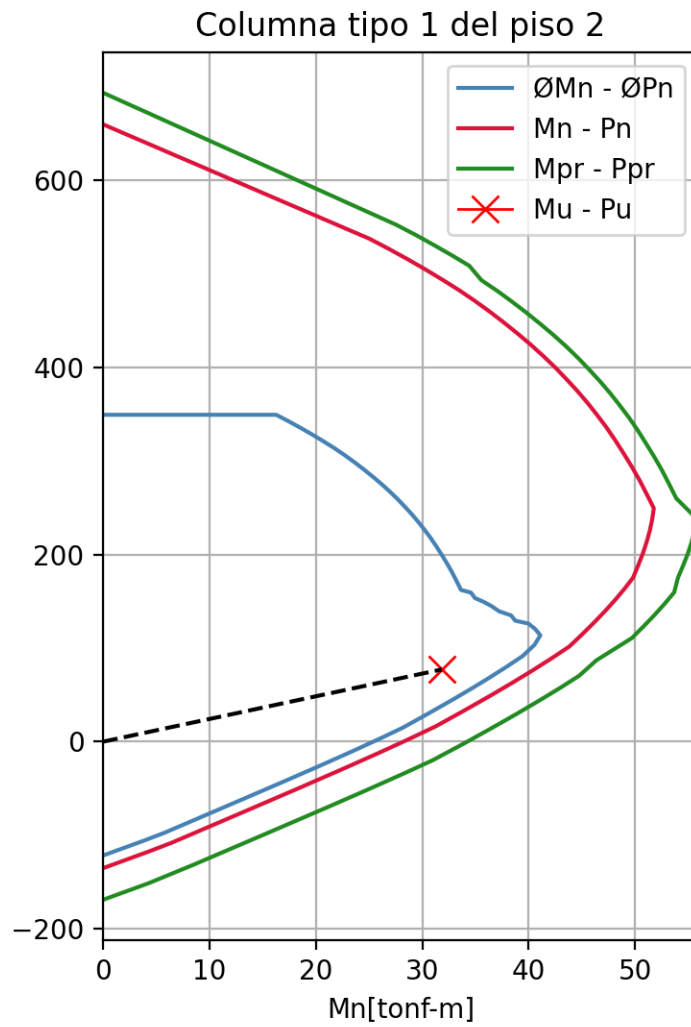


Figura 44: Curva de interacción de la columna del tipo 1 del piso 2.

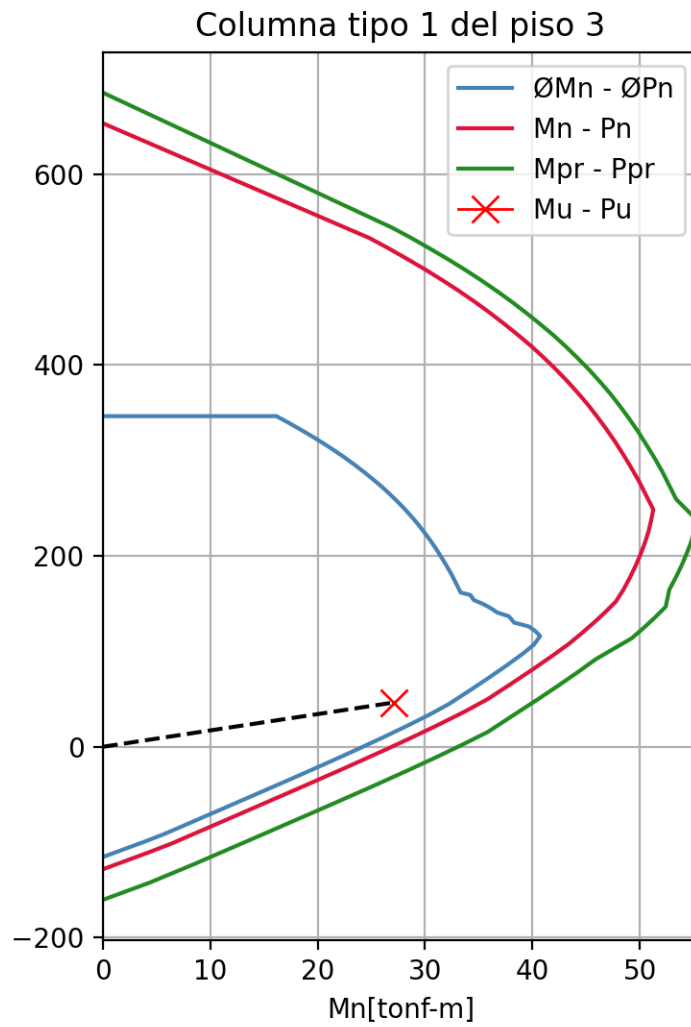


Figura 45: Curva de interacción de la columna del tipo 1 del piso 3.

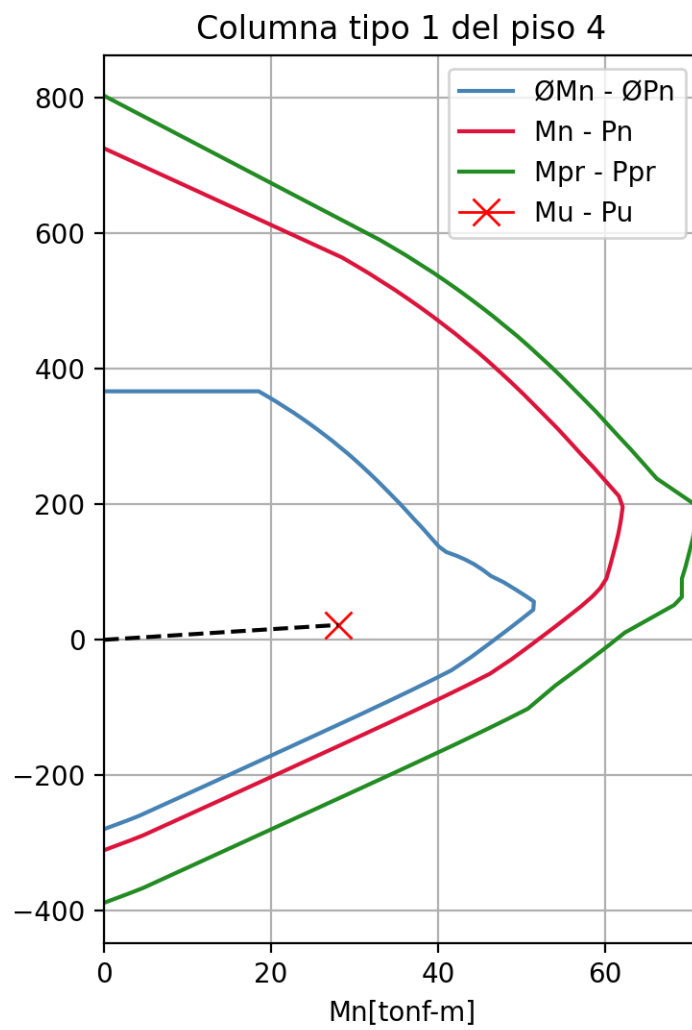


Figura 46: Curva de interacción de la columna del tipo 1 del piso 4.

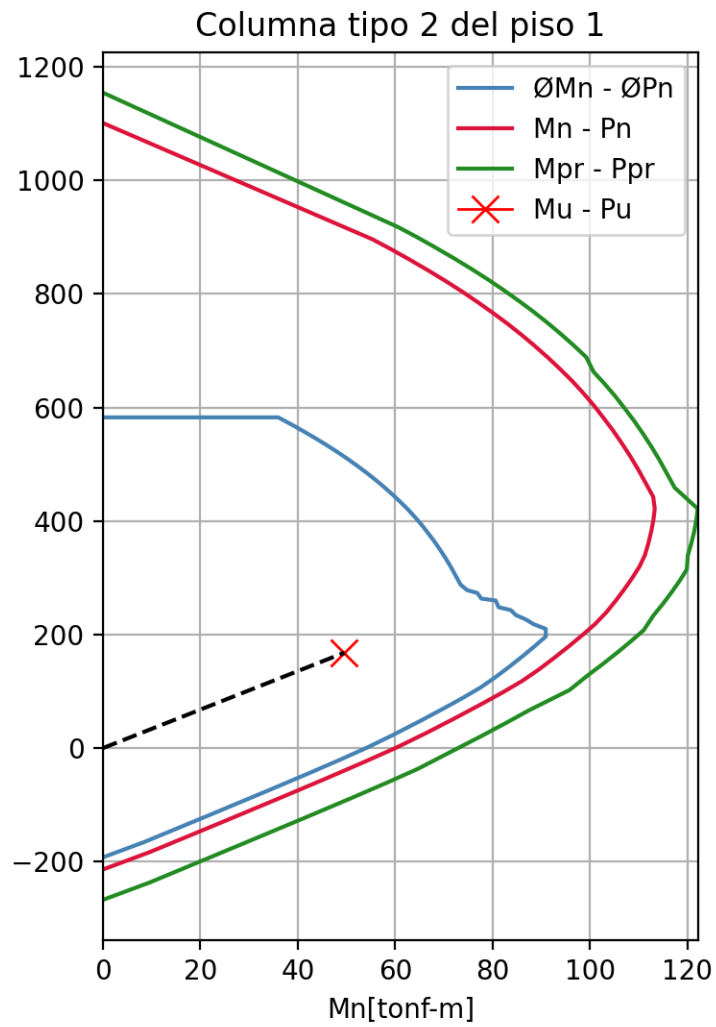


Figura 47: Curva de interacción de la columna del tipo 2 del piso 1.

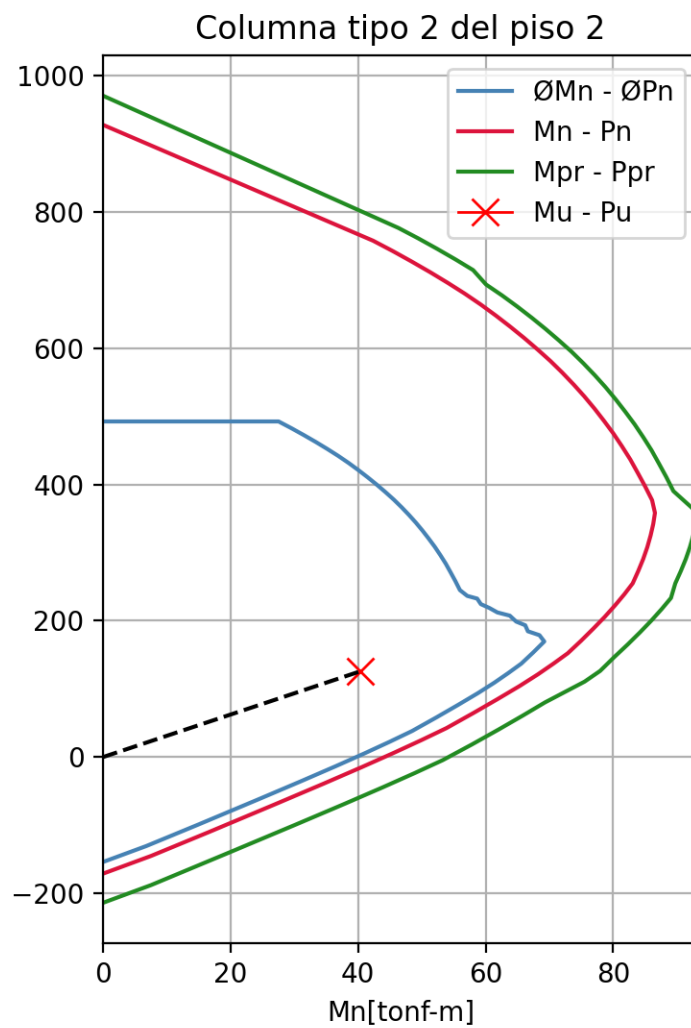


Figura 48: Curva de interacción de la columna del tipo 2 del piso 2.

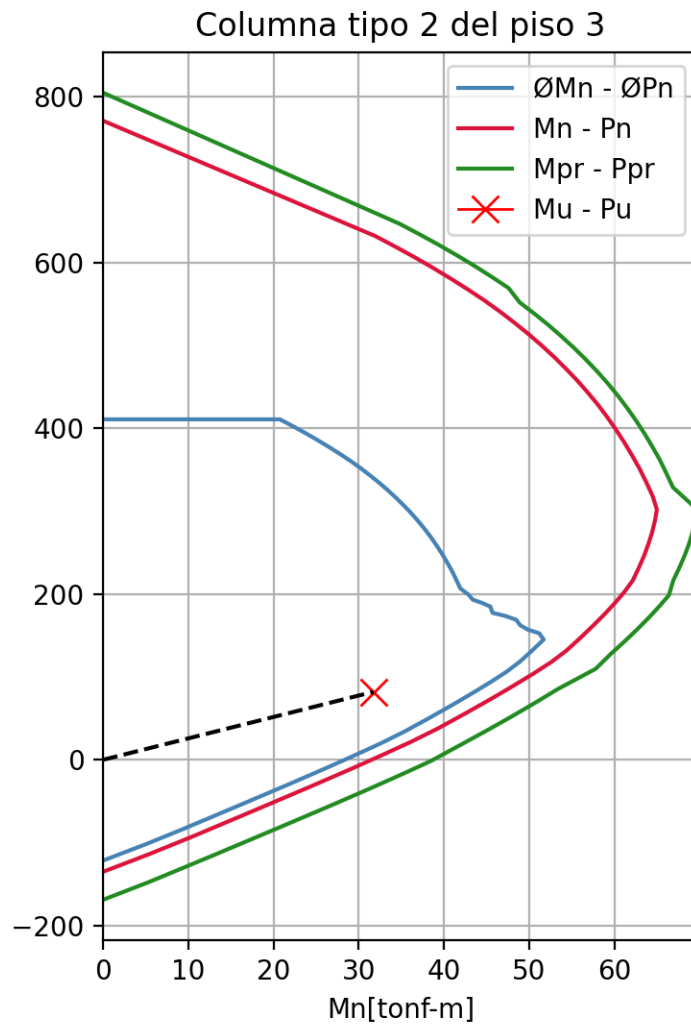


Figura 49: Curva de interacción de la columna del tipo 2 del piso 3.

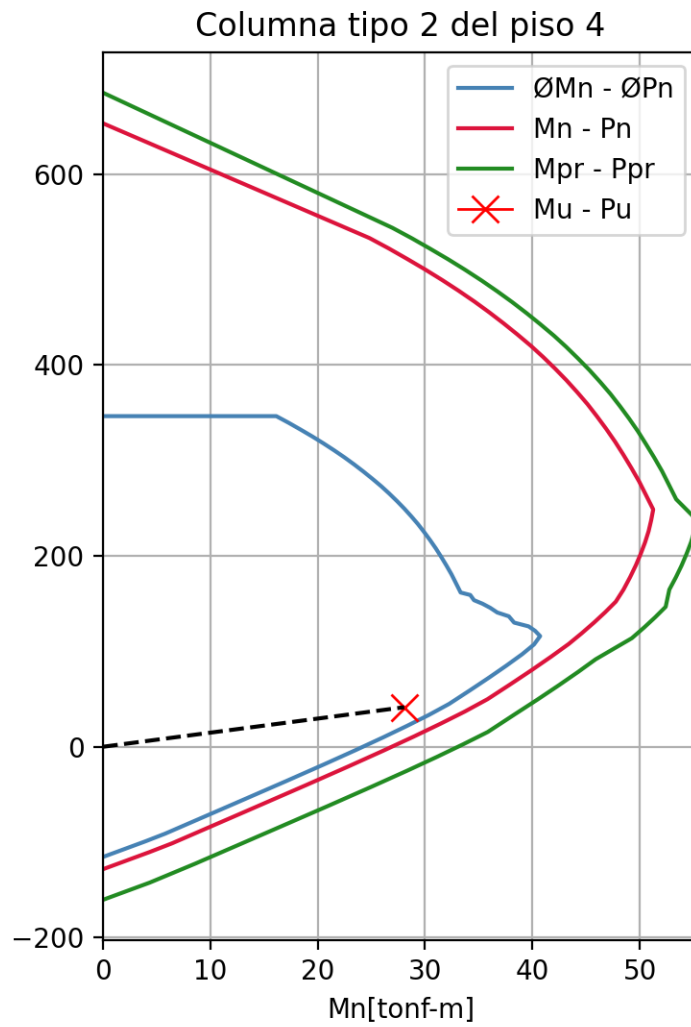


Figura 50: Curva de interacción de la columna del tipo 1 del piso 1.

Anexo C – Cálculo de expresiones algebraicas

C.1. Flexión simple

C.1.1. Refuerzo de tracción

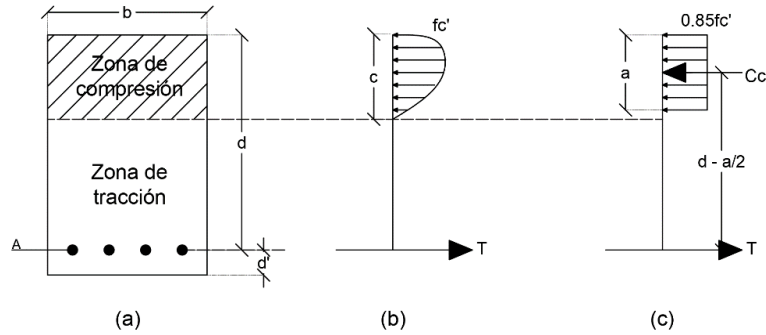


Figura 51: Simplificación de la distribución de esfuerzos en una sección reforzada a tracción.

Para calcular el refuerzo a tracción, se desprecia el aporte de resistencia a tracción del hormigón y la resistencia a compresión se distribuye uniformemente de acuerdo con la simplificación del punto 22.2.2.4.1 con los valores de β_1 de la tabla 22.2.2.4.3 del código ACI318S-14. De esta forma, el aporte de resistencia a compresión del hormigón es:

$$C_c = 0.85f'_c ab_w$$

Por otro lado, la resistencia del acero en tracción es:

$$T = A_s f_y$$

Para comenzar, se realiza un equilibrio de fuerzas en la sección asumiendo que $T = C_c$, y se despeja 'a':

$$a = \frac{A_s f_y}{0.85f'_c b_w} \quad (C.1)$$

Y luego se realiza la sumatoria de momentos en C_c de acuerdo con la Figura 51:

$$\sum M_{C_c} = 0 \leftrightarrow M_n = T \left(d - \frac{a}{2} \right)$$

Se sustituye los valores de 'a' de la (C.1), $T = A_s f_y$ y $M_n = \frac{M_u}{\phi}$ en esta última expresión:

$$\frac{M_u}{\phi} = A_s \cdot f_y \left(d - \frac{1}{2} \cdot \frac{A_s f_y}{0.85 f'_c \cdot b_w} \right)$$

Esta última expresión se multiplica por $\frac{1}{0.85 f'_c \cdot b_w d^2}$:

$$\frac{M_u}{\phi 0.85 f'_c \cdot b_w d^2} = \frac{A_s f_y}{0.85 f'_c b_w d^2} \left(d - \frac{1}{2} \cdot \frac{A_s f_y}{0.85 f'_c b_w} \right) \quad (C.2)$$

Como esta expresión resulta muy grande, se definen μ y ω , que corresponden al momento reducido y cuantía mecánica respectivamente:

$$\mu = \frac{M_u}{\phi 0.85 f'_c b_w d^2} \quad (C.3)$$

$$\omega = \frac{A_s f_y}{0.85 f'_c b_w d} \quad (C.4)$$

Se reemplazan μ y ω , provenientes de las ecuaciones (C.3) y (C.4) en la ecuación (C.2):

$$\mu = \omega \left(1 - \frac{\omega}{2} \right) \quad (C.5)$$

Cuya única raíz viable es:

$$\omega = 1 - \sqrt{1 - 2\mu} \quad (\text{C.6})$$

Luego, se reemplaza la expresión de ‘a’ por ‘a’, proveniente de la ecuación (C.1) en la ecuación (C.4) y el valor de ‘a’ se reemplaza a su vez por $a = c\beta_1$, quedando así:

$$d = \frac{c\beta_1}{\omega}$$

Cuyo recíproco multiplicado por ‘c’ resulta:

$$\frac{c}{d} = \frac{\omega}{\beta_1}$$

Y se define $\xi = \frac{c}{d}$, ecuación que representa la relación entre la profundidad de la línea neutra ‘c’ y la distancia de la sección que aporta resistencia ‘d’. Al ser sustituida en la expresión anterior queda:

$$\xi = \frac{\omega}{\beta_1}$$

Para garantizar la ductilidad, se fija la deformación unitaria del acero en 0.005 y considerando la deformación del hormigón como 0.003, tomando como referencia la figura Figura 52b, se crea una relación lineal donde

$$\frac{c}{d} = \frac{0.003}{0.003 + 0.005} = \frac{3}{8}$$

por lo que,

$$\xi_{lim} = \frac{3}{8}$$

Luego, al despejar ω de $\xi = \frac{\omega}{\beta_1}$ y considerar el valor de ξ_{lim} , se obtiene que:

$$\omega_{lim} = \frac{3}{8} \beta_1$$

Ecuación que sustituida en la ecuación (C.5) resulta:

$$\mu_{lim} = \frac{3}{8} \beta_1 \left(1 - \frac{3}{16} \beta_1 \right) \quad (C.7)$$

Si $\mu > \mu_{lim}$, entonces se debe considerar armadura de compresión para que $\varepsilon_t \geq 0.005$.

C.1.2. Refuerzo de compresión

El refuerzo a compresión consta de barras de acero longitudinales ubicadas sobre la profundidad de línea neutra 'c', esto es, en la zona de compresión.

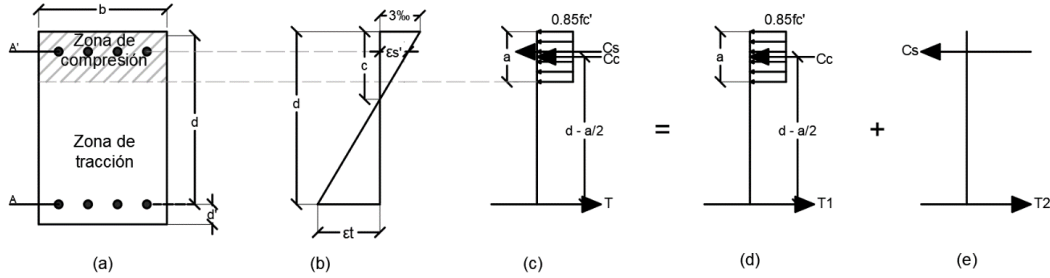


Figura 52: Distribución de esfuerzos en una sección doblemente reforzada.

Se definen $C_s = A' f_s'$, $\alpha = \frac{f_s'}{f_y}$, $T_1 = (A - \alpha A') f_y$, y, $T_2 = A' f_s'$, para luego realizar la sumatoria de fuerzas de acuerdo con la Figura 52(d) y se despeja 'a':

$$\sum F_y = 0 \Leftrightarrow C_c = T_1 \Rightarrow 0.85 f_c' b_w a = (A - \alpha A') f_y \Rightarrow a = \frac{(A - \alpha A') f_y}{0.85 f_c' b_w}$$

Se realiza sumatoria de momentos de acuerdo con la Figura 52(c) asumiendo la expresión anterior $C_c = T_1$, quedando así el momento nominal:

$$M_n = T_1 \left(d - \frac{a}{2} \right) + T_2 (d - d') \quad (C.8)$$

Se multiplica por $\frac{1}{0.85f'_c \cdot b_w d^2}$, se reemplaza M_n por $\frac{M_u}{\phi}$, se simplifica la expresión $(d - d')$ por d , y, se sustituyen los valores de 'a', T1 y T2:

$$\begin{aligned} \frac{M_u}{\phi 0.85f'_c \cdot b d^2} &= \frac{(A - \alpha A') f_y}{0.85f'_c \cdot b d} \left(1 - \frac{1}{2} \cdot \frac{(A - \alpha A') f_y}{0.85f'_c \cdot b d} \right) \\ &+ \frac{\alpha A' f_y}{0.85f'_c \cdot b d} \left(1 - \frac{d'}{d} \right) \end{aligned} \quad (C.9)$$

Se utiliza $a = a_{lim}$ y se calcula ω_{lim} de acuerdo con $\omega = \frac{a}{d}$, se define $\delta' = \frac{d'}{d}$, y, se define ω' para la armadura de compresión:

$$\omega_{lim} = \frac{(A - \alpha A') f_y}{0.85f'_c \cdot b_w d}$$

$$\omega' = \frac{\alpha A' f_y}{0.85f'_c \cdot b_w d}$$

$$\Rightarrow \mu = \omega_{lim} \left(1 - \frac{1}{2} \omega_{lim} \right) + \alpha \cdot \omega' (1 - \delta')$$

Donde al remplazar el primer término por μ_{lim} , de acuerdo con la ecuación (C.5)

$$\mu = \mu_{lim} + \alpha \cdot \omega' (1 - \delta')$$

Se despeja ω' :

$$\omega' = \frac{\mu - \mu_{lim}}{\alpha(1 - \delta')}$$

Se replantea ω_{lim} , en función de ω y ω' :

$$\omega_{lim} = \omega - \alpha \cdot \omega'$$

Y finalmente se despeja ω :

$$\omega = \omega_{lim} + \alpha \cdot \omega' \quad (C.10)$$

Es importante mencionar que si $A' = 0$, se recuperan las expresiones provenientes de una armadura con refuerzo a tracción únicamente.

Los momentos probables en las vigas se calculan en cambio, aumentando el valor de f_y 1.25 veces y para simplificarlo se utilizará una función para verificar flexo-compresión con carga axial cero y el f_y modificado.

C.2. Flexo-compresión

C.2.1. Cálculo de cargas y momentos nominales

La flexo-compresión, también llamada flexión compuesta, es la combinación de esfuerzos de flexión con esfuerzos axiales, dado un momento y carga solicitados.

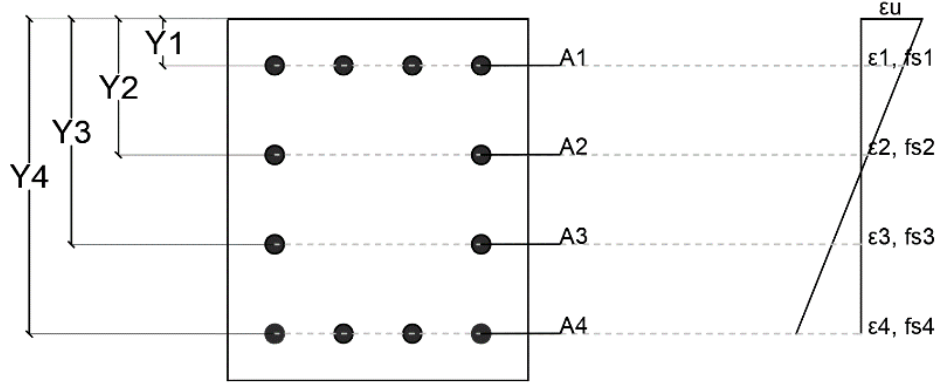


Figura 53: Sección transversal de un posible elemento de hormigón armado.

La sección transversal de un elemento tipo analizado, ya sea viga o columna, se puede ver en la Figura 53, en esta se observa cómo varía la tensión admisible del acero de acuerdo con si deformación unitaria.

Recordando que la deformación del acero es:

$$\varepsilon_s = \frac{c - Y_i}{c} \cdot \varepsilon_{ty}$$

Y que el valor de f_s está limitado por el comportamiento elastoplástico, tal como en la Figura 7, tomando como límite $\varepsilon_{ty} = 0.002$, como se menciona en el punto 21.2.2.1 del código ACI 318S-14 para aceros de grado 420, dando origen a la expresión:

$$f_s = \begin{cases} E_s \cdot \varepsilon_s, & si \quad |\varepsilon_s| < \varepsilon_{ty} \\ f_{ty} \cdot \frac{\varepsilon_s}{|\varepsilon_s|}, & si \quad |\varepsilon_s| \geq \varepsilon_{ty} \end{cases}$$

La fuerza en cada nivel de área se calcula como:

$$P_s = \sum P_{si} = \sum f_{si} A_i$$

La carga nominal aportada por el hormigón:

$$P_c = 0,85\beta_1 f'_c b_w c$$

La carga nominal total:

$$P_n = P_c + P_s$$

El momento nominal aportado por el hormigón:

$$M_c = P_c \left(\frac{h}{2} - \frac{0,85c}{2} \right)$$

El momento nominal aportado por las barras de acero:

$$M_s = \sum P_{si} \left(\frac{h}{2} - Y_i \right)$$

El momento nominal total:

$$M_n = M_c + M_s$$

C.2.2. Curvas de interacción

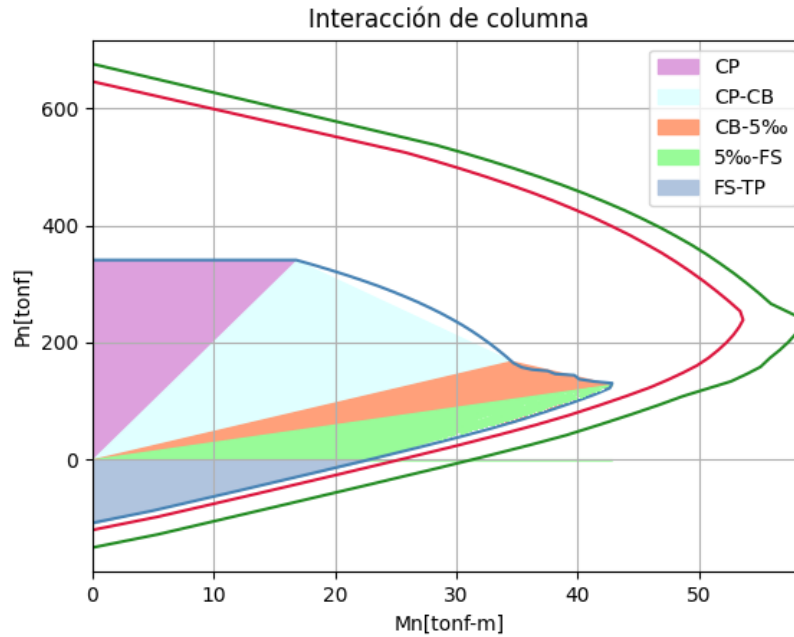


Figura 54: Zonas del diagrama de interacción en una columna.

En el gráfico de la Figura 54 se pueden distinguir 3 curvas, la curva de diseño, con el factor de reducción aplicado (azul), la curva teórica o nominal (roja) y la de cargas y momentos probables (verde), cuya aplicación servirá para el cálculo de corte por capacidad y cuyas coordenadas son los momentos M en el eje de las abscisas y las cargas P en el eje de las ordenadas.

En las curvas de interacción se distinguen 5 puntos en cuya traza se reparten 4 zonas de transición. Los puntos varían de acuerdo con la profundidad de la línea neutra y reflejan la capacidad de la sección de la columna de acuerdo con su excentricidad, dicho de otra manera, la relación entre momento último y carga última, $e = \frac{M_u}{P_u}$. Estos puntos son: compresión pura, condición balanceada al 5‰ ($\varepsilon_t = 0.005$), flexión simple y tracción pura. Cada uno de estos puntos se ajustan a una determinada profundidad de la línea neutra. A continuación, se sitúan estos 5 puntos:

- Compresión pura: La Resistencia a la compresión nominal simple corresponde al 80% de la resistencia a la compresión nominal, de acuerdo con el punto b de la tabla 22.4.2.1 del ACI 318-14. En este caso, para secciones rectangulares y con estribos:

$$P_u = \phi P_{n_{MAX}} = 0.80\phi \left(0.85f'_c(A_g - A_s) \right) + A_s \cdot f_y$$

- Condición balanceada: En condición balanceada, la profundidad de línea neutra es:

$$c = \frac{3}{5}d.$$

- Deformación Unitaria del 5‰: la profundidad de la línea neutra para este punto se ubica en $c = \frac{3}{8}d$.
- Flexión Simple: La flexión simple ocurre cuando el valor de P_n es nulo.
- Tracción pura: En tracción pura, la profundidad de la línea neutra es cero y el momento nominal es nulo, su carga nominal se calcula como $P_n = -A_T \cdot f_y$.

Con el conjunto de puntos, contruidos a partir de la descripción anterior, se puede construir la curva roja, que representa las cargas y momentos nominales, luego, la curva azul se puede construir multiplicando los esfuerzos nominales por el factor de reducción y la curva verde, que representa los esfuerzos probables, aumentando el valor de f_y en el cálculo por 1.25 veces.

C.3. Factor de corrección de área

En cada iteración donde se verifica el drift, se corrigen las secciones para conseguir una estructura que cumpla con un drift máximo del 2‰. Se define el factor de corrección de área, ψ , con el que se aumentan o disminuyen las dimensiones actuales de las secciones de las vigas y columnas, según si se requiere una estructura más rígida o flexible respectivamente y conservando la razón de sus dimensiones, hasta que el factor tienda a 1.

Para definir el factor ψ , primero se define para ámbitos de la demostración un factor r , producto de la relación entre los drifts (máximo y admisible) de la estructura:

$$drift_{adm} = \frac{0.002}{FS}$$

$$r = \frac{drift_{max}}{drift_{adm}}$$

Donde:

- $drift_{adm}$: Drift admisible.
- $drift_{max}$: Drift máximo de la iteración.
- FS : Factor de seguridad por omisión de los efectos de torsión, equivalente a 1.2.
- r : Factor de corrección de inercia.

Luego, se define la inercia de una sección cualquiera únicamente en función del alto, para lo cual se define un factor ρ que conserva la proporción entre las dimensiones de la sección, haciendo depender al ancho del alto:

$$I = \frac{b \cdot h^3}{12}$$

$$b(h) = \rho \cdot h$$

$$I = \rho \frac{h^4}{12}$$

Finalmente, para deducir el factor ψ , se parte de la hipótesis que se corrige la inercia de las secciones, la que contempla las dimensiones ya corregidas:

$$r \cdot I = \rho \frac{(\psi h)^4}{12}$$

$$\sqrt[4]{\frac{12}{\rho} \cdot r \cdot I} = \psi h$$

$$\sqrt[4]{r} \cdot \sqrt[4]{\frac{12}{\rho}} \cdot I = \psi \cdot h$$

$$\psi = \sqrt[4]{r}$$

Donde:

- b : Ancho de la sección.
- h : Alto de la sección.
- I : Inercia de la sección en torno al eje local X.
- ρ : Razón entre el ancho y el alto de la sección.
- ψ : Factor de corrección de área.

Anexo D – Código Fuente β -beam

```
MERYTEST := 0:
import tracemalloc
tracemalloc.start()
from math import pi as  $\pi$ , sin
from numpy import array, zeros, round as mround
from numpy.linalg import inv, solve
import matplotlib.pyplot as pete

def line(): print('---- - - - -')

def varsC(dirl=dir()):
i = 0
lista = []
ipyvars = ['In', 'Out', 'exit', 'get_ipython', 'quit']
for j in dirl:
if j[0] == '_': continue
elif j in ipyvars: continue
i += 1
lista.append((i, j))
return lista

def period():
"""formulas del paper en base al ejemplo de la tesis"""
Sc = 67500*8*0.01**4
Sb = 160000*6*0.01**4
B = 1.5*3
H = 1.2*2
W = 0 # FALTA
T = 0.374*H**1.1*W/(B**0.3*Sc**0.31*Sb**0.35)
print(0.09*H/3**0.5)
print(0.075*H**0.75)
print(0.0466*H**0.9)
print(T)
print(0.879*H**0.43*3**0.65*W**0.18/B)
print(0.475*H**0.547*3**0.103/B**0.271)
print(0.428*H**0.545/B**0.185)
print(0.28*H**0.54)
print(0.012*H)
return

p = print

# lectura de datos

def lectura():
"""Lee los parametros definidos por el usuario desde un txt. Si\
se omite el valor, se autocompleta con los valores por defecto."""

with open('../IO/input.txt', encoding='utf-8') as in_text:
in_temp = in_text.readlines()

default = {
'numero de pisos': '2',
'numero de vanos': '3',
'numero de marcos': '4',
'altura de piso': '240',
'distancia entre columnas': '300',
'distancia entre marcos': '300',
'espesor de losas': '15',

'modulo de poisson  $\nu$ ': '0.25',
'resistencia fc hormigon': '19.5',
'densidad del hormigon': '2500',
'sobrecarga de uso': '500',

'zona sismica': '3',
'tipo de suelo': 'E',
'categoria de edificacion': 'II',
'factor de modificacion R': '7',
'porcentaje de sobrecarga': '0.25',

'ancho de columnas': '90',
'ancho de vigas': '40',
'altura de vigas': '90',
}

user_op = [] # option names from txt file
user_in = [] # values given by the user

for i in in_temp:
if i.startswith('\n') == False:
user_op.append(i.partition(':')[0].strip())
user_in.append(i.partition(':')[2].strip())
print(user_in)
```

```

del in_temp
if user_op != list(default.keys()): raise Exception('wena')

for i in range(len(user_in)):
    if user_in[i] == '':
        user_in[i] = list(default.values())[i]
    elif user_in[i].isalpha() == False:
        user_in[i] = float(user_in[i])
for i in range(7): user_in[i] = int(user_in[i])
print(user_in)
print(user_op)
return

lectura()

class Grilla():
    """Base para definir las dimensiones una estructura."""
    def __init__(grid, grilla):
        grid.grilla = grilla
        grid.dimXY = grid.dimensions()
        grid.dim = {'x': grid.dimXY[1], 'y': grid.dimXY[0]}
        grid.nodos = grid.nodos()
        grid.max_barras = (grid.dim['x'] - 1)*grid.dim['y'] \
            + grid.dim['y']*grid.dim['x']
        grid.K = zeros((3*grid.nodos, 3*grid.nodos)) # rigidez del sistema
        grid.r = 0 # desplazamientos
        grid.drift = {}
        grid.annainfo = {}
        grid.VERB = True # true = verbose on
        return
    def clean(grid):
        grid.annainfo = {
            'esfuerzos': {
                'head': [
                    ['perfil', 'combo', 'nodo', 'axial', 'corte', 'momento'],
                    ['[kg]', '[kg]', '[kg-m]']
                ],
                'body': []
            },
            'delta-nodos': {
                'head': [
                    ['combo', 'nodo', 'X', 'Y', 'Giro'],
                    ['[mm]', '[mm]', '[rad]']
                ],
                'body': []
            },
            'drifts': {
                'head': [
                    ['combo', 'nivel', 'δmax', 'δmin', 'drift'],
                    ['[mm]', '[mm]', '[%]']
                ],
                'body': []
            }
        }
        return
    def dimensions(grid):
        """Retorna la tupla (nodos en Y, nodos en X)."""
        return len(grid.grilla), len(grid.grilla[0])
    def nodos(grid):
        """Retorna la cantidad de nodos."""
        return grid.dimXY[0]*grid.dimXY[1]
    def maxElements(grid):
        return
    def dibujar(grid):
        marco2, ejes2 = pete.subplots(1, 1, figsize=(8.0,4.0))
        ejes2.set_xticks([i*b_losas for i in range(n_vanos + 1)])
        ejes2.set_yticks([i*h_pisos for i in range(n_pisos + 1)])
        for m in miembro:
            if not m: continue
            ejes2.plot([m.x0, m.x1], [m.y0, m.y1], 'k', linewidth=2)
        marco3 = pete.figure()
        ejes3 = pete.axes(projection='3d')
        ejes3.grid(False)
        ejes3.set_xticks([i*b_losas/100 for i in range(n_vanos + 1)])
        ejes3.set_yticks([i*d_losas/100 for i in range(n_marcos)])
        ejes3.set_zticks([])

        z = 0
        for marco in range(1, n_marcos + 1):
            for m in miembro:
                if not m: continue
                if m.name == 'COL' and marco < n_marcos:
                    ejes3.plot3D([m.x1/100, m.x1/100], [z/100, (z + d_losas)/100], [m.y1/100, m.y1/100],
                        'r', linewidth=0.5)
                if marco == n_marcos - 1:

```

```

        ejes3.plot3D([m.x0/100, m.x1/100], [z/100, z/100], [m.y0/100, m.y1/100], 'k',
linewidth=2)
    else:
        ejes3.plot3D([m.x0/100, m.x1/100], [z/100, z/100], [m.y0/100, m.y1/100], 'b',
linewidth=0.5)
        z += d_losas

        ejes3.view_init(3, -45)
        marco3.savefig(f'_marco_vista_a.svg')
        ejes3.view_init(6, -45)
        marco3.savefig(f'_marco_vista_b.svg')
        ejes3.view_init(12, -12)
        marco3.savefig(f'_marco_vista_c.svg')
        #ejes3.view_init(60, -72)
        #marco3.savefig(f'_marco_vista_d.svg')
        return

TEST = 3
if TEST == 3:
    struct = Grilla([
        [1,2,3,4],
        [5,6,7,8],
        [9,10,11,12],
        [13,14,15,16],
    ])
elif TEST == 2:
    struct = Grilla([
        [1,2,3,4],
        [5,6,7,8],
    ])
elif TEST == 1:
    struct = Grilla([[1,2]])

vars(struct)

# test - Input desde archivo txt
zona = '3'
categoria = 'II'
suelo = 'E'
R = 7

if TEST == 3:
    h_pisos = 300 # [cm] pero despues sera en cm y despues en varias unidades
    n_pisos = 4
    n_vanos = 3
elif TEST == 1:
    h_pisos = 240 # [cm] pero despues sera en cm y despues en varias unidades
    n_pisos = 1
    n_vanos = 1

n_marcos = 4

if TEST == 3:
    d_losas = 700 # [cm]
    b_losas = 700 # [cm]
if TEST == 1:
    d_losas = 300 # [cm]
    b_losas = 300 # [cm]

t_losas = 15 # [cm]
Q_L = 500/100**2 # [kg/cm2]

b_VIG = 40 # [cm]
h_VIG = 90 # [cm]
b_COL = 90 # [cm]
h_COL = 90 # [cm]
x = 0
v = 0.25
γ = 2500
fc = 25

# definicion de elementos

class barra():
    # agregar rigidos
    def __init__(bar, nodos, i, j, x0, y0, x1, y1, b, h, x, v, γ, fc):
        """Define los parametros del perfil y material de un elemento 'barra'."""

        #### ## input ## ####
        bar.i = i # nodo A
        bar.j = j # nodo B
        bar.x0 = x0 # [cm]
        bar.y0 = y0 # [cm]
        bar.x1 = x1 # [cm]

```

```

bar.y1 = y1 # [cm]
bar.b = b # [cm]
bar.h = h # [cm]
bar.x = x # factor de forma de la seccion
bar.v = v # modulo de poisson
bar.y = y/100**3 # [kg/m3] -> [kg/cm3] - densidad
bar.fc = fc # [MPa] - resistencia en compresion

if x0 != x1:
    bar.name = 'VIGA'
else:
    bar.name = 'COL'

#### ## internals ## ####
bar.GdL(i, j, nodos) # grados de libertad

bar.L = ((x1 - x0)**2 + (y1 - y0)**2)**0.5 # [cm]
bar.s = (y1 - y0)/bar.L # senos
bar.c = (x1 - x0)/bar.L # coseno
bar.A = b*h # [cm2]
bar.I = b*h**3/12 # [cm4]
bar.E = y**1.5*0.043*fc**0.5/9.80665/0.01 # [kg/mm2]/0.01 = [kg/cm2]
bar.G = bar.E/(2*(1+v)) # modulo de cizalle
bar.β = 6*bar.E*bar.I*x/(bar.G*bar.A*bar.L**2)

bar.rigidez_local(bar.L, bar.A, bar.I, bar.E, bar.β)
bar.compatibilidad_geometrica(bar.s, bar.c, bar.L)

bar.K = bar.a.T@bar.k@bar.a # rigidez global

bar.dst_loads = [] # (a, b, [kg/cm]) - cargas distribuidas
bar.jnt_loads = [] # [kg] [kg-cm] - cargas nodales
bar.caso = {} # suma de todas las cargas basicas por cada caso
bar.combo = {} # vectores de cada combinacion de casos
bar.q_caso = {}
bar.q_combo = {}
return

def GdL(bar, a, b, nodosEstruc, tipo=''):
    """Prepara la matriz de transformacion de cada barra."""
    bar.T = zeros([6, 3*nodosEstruc])
    if a > nodosEstruc or b > nodosEstruc:
        print('index error')
        return
    if a < 0 or b < 0:
        print('index error')
        return
    if a != 0:
        bar.T[0, 3*a - 3] = 1
        bar.T[1, 3*a - 2] = 1
        bar.T[2, 3*a - 1] = 1
    if b != 0:
        bar.T[3, 3*b - 3] = 1
        bar.T[4, 3*b - 2] = 1
        bar.T[5, 3*b - 1] = 1
    return

def compatibilidad_geometrica(bar, s, c, L):
    bar.a = array([
        [-s/L, c/L, 1, s/L, -c/L, 0],
        [-s/L, c/L, 0, s/L, -c/L, 1],
        [c, s, 0, -c, -s, 0]
    ])
    return

def rigidez_local(bar, L, A, I, E, β):
    k11 = 2*E*I*(2 + β)/(L*(1 + 2*β))
    k12 = 2*E*I*(1 - β)/(L*(1 + 2*β))
    k21 = k12
    k22 = k11
    k33 = A*E/L
    bar.k = array([
        [k11, k12, 0],
        [k21, k22, 0],
        [0, 0, k33]
    ])
    return

def addDistF(bar, a, b, fa, fb, coord='Y'):
    """Ingresa una fuerza distribuida desde 'a' hasta 'b'."""
    pass

def sigma(bar, r_global): # sin uso - borrar
    # σ = k@ε
    # ε = a@r_i

```

```

# r_i = T@r
# σ = k@a@T@r
bar.σ = bar.k@bar.a@bar.T@r_global # esfuerzos

def dist2joint(bar): pass

def calcular_peso_propio(bar):
    """Devuelve el peso propio de la barra en kg/cm. La tupla\
    contiene tramo inicial y final donde la carga no es constante."""
    γ = bar.γ
    b = bar.b
    h = bar.h
    bar.dst_loads.append((_pp-bar, 'D', (0, 0, γ*b*h)))
    return

def case_gen(bar, accion='generar'):
    """Genera las categorías de cargas básicas de la barra."""
    if accion == 'generar':
        case_keys = ['D', 'L']
        bar.q_caso = {} # rect dist loads
        for i in case_keys: bar.q_caso.update({i: 0})
        bar.caso = {} # joint loads
        for i in case_keys: bar.caso.update({i: zeros(6)})
    return

def update(bar):
    bar.I = bar.b*bar.h**3/12
    bar.A = bar.b*bar.h
    bar.rigidez_local(bar.L, bar.A, bar.I, bar.E, bar.β)
    bar.K = bar.a.T@bar.k@bar.a # rigidez global
    return

def calculate_el_piso(): pass

r = 0 # desplazamientos
ε = 0 # deformaciones
σ = 0 # esfuerzos en coordenadas locales [Ma, Mb, F]
sigma = 0 # esfuerzos en coordenadas globales [Na, Va, Ma, Nb, Vb, Mb]
v = 0 # esfuerzo de corte
hayPP = False # indexador caso PesoPropio - actualizar en el futuro
losa = {'D': [], 'L': []} # indexador casos losa
caso: {} # casos de cargas básicas

# despues agregar tramos rigidos

# definicion geometrica de cada elemento

def armar_estructura(test, NODOS):
    """Genera todos los elementos 'barra' de la estructura."""
    barras = []
    if test == 3:
        m1 = barra(NODOS, 0, 1, 0, 0, 0, 300, b_COL, h_COL, x, v, γ, fc)
        m2 = barra(NODOS, 0, 2, 700, 0, 700, 300, b_COL, h_COL, x, v, γ, fc)
        m3 = barra(NODOS, 0, 3, 1400, 0, 1400, 300, b_COL, h_COL, x, v, γ, fc)
        m4 = barra(NODOS, 0, 4, 2100, 0, 2100, 300, b_COL, h_COL, x, v, γ, fc)
        m5 = barra(NODOS, 1, 5, 0, 300, 0, 600, b_COL, h_COL, x, v, γ, fc)
        m6 = barra(NODOS, 2, 6, 700, 300, 700, 600, b_COL, h_COL, x, v, γ, fc)
        m7 = barra(NODOS, 3, 7, 1400, 300, 1400, 600, b_COL, h_COL, x, v, γ, fc)
        m8 = barra(NODOS, 4, 8, 2100, 300, 2100, 600, b_COL, h_COL, x, v, γ, fc)
        m9 = barra(NODOS, 5, 9, 0, 600, 0, 900, b_COL, h_COL, x, v, γ, fc)
        m10 = barra(NODOS, 6, 10, 700, 600, 700, 900, b_COL, h_COL, x, v, γ, fc)
        m11 = barra(NODOS, 7, 11, 1400, 600, 1400, 900, b_COL, h_COL, x, v, γ, fc)
        m12 = barra(NODOS, 8, 12, 2100, 600, 2100, 900, b_COL, h_COL, x, v, γ, fc)
        m13 = barra(NODOS, 9, 13, 0, 900, 0, 1200, b_COL, h_COL, x, v, γ, fc)
        m14 = barra(NODOS, 10, 14, 700, 900, 700, 1200, b_COL, h_COL, x, v, γ, fc)
        m15 = barra(NODOS, 11, 15, 1400, 900, 1400, 1200, b_COL, h_COL, x, v, γ, fc)
        m16 = barra(NODOS, 12, 16, 2100, 900, 2100, 1200, b_COL, h_COL, x, v, γ, fc)
        m17 = barra(NODOS, 1, 2, 0, 300, 700, 300, b_VIG, h_VIG, x, v, γ, fc)
        m18 = barra(NODOS, 2, 3, 700, 300, 1400, 300, b_VIG, h_VIG, x, v, γ, fc)
        m19 = barra(NODOS, 3, 4, 1400, 300, 2100, 300, b_VIG, h_VIG, x, v, γ, fc)
        m20 = barra(NODOS, 5, 6, 0, 600, 700, 600, b_VIG, h_VIG, x, v, γ, fc)
        m21 = barra(NODOS, 6, 7, 700, 600, 1400, 600, b_VIG, h_VIG, x, v, γ, fc)
        m22 = barra(NODOS, 7, 8, 1400, 600, 2100, 600, b_VIG, h_VIG, x, v, γ, fc)
        m23 = barra(NODOS, 9, 10, 0, 900, 700, 900, b_VIG, h_VIG, x, v, γ, fc)
        m24 = barra(NODOS, 10, 11, 700, 900, 1400, 900, b_VIG, h_VIG, x, v, γ, fc)
        m25 = barra(NODOS, 11, 12, 1400, 900, 2100, 900, b_VIG, h_VIG, x, v, γ, fc)

```

```

m26 = barra(NODOS, 13, 14, 0, 1200, 700, 1200, b_VIG, h_VIG, x, v, γ, fc)
m27 = barra(NODOS, 14, 15, 700, 1200, 1400, 1200, b_VIG, h_VIG, x, v, γ, fc)
m28 = barra(NODOS, 15, 16, 1400, 1200, 2100, 1200, b_VIG, h_VIG, x, v, γ, fc)
elif test == 2: # tesis
    m1 = barra(NODOS, 0, 1, 0, 0, 0, 120, b_COL, h_COL, x, v, γ, fc)
    m2 = barra(NODOS, 0, 2, 150, 0, 150, 120, b_COL, h_COL, x, v, γ, fc)
    m3 = barra(NODOS, 0, 3, 300, 0, 300, 120, b_COL, h_COL, x, v, γ, fc)
    m4 = barra(NODOS, 0, 4, 450, 0, 450, 120, b_COL, h_COL, x, v, γ, fc)
    m5 = barra(NODOS, 1, 5, 0, 120, 0, 240, b_COL, h_COL, x, v, γ, fc)
    m6 = barra(NODOS, 2, 6, 150, 120, 150, 240, b_COL, h_COL, x, v, γ, fc)
    m7 = barra(NODOS, 3, 7, 300, 120, 300, 240, b_COL, h_COL, x, v, γ, fc)
    m8 = barra(NODOS, 4, 8, 450, 120, 450, 240, b_COL, h_COL, x, v, γ, fc)
    m9 = barra(NODOS, 1, 2, 0, 120, 150, 120, b_VIG, h_VIG, x, v, γ, fc)
    m10 = barra(NODOS, 2, 3, 150, 120, 300, 120, b_VIG, h_VIG, x, v, γ, fc)
    m11 = barra(NODOS, 3, 4, 300, 120, 450, 120, b_VIG, h_VIG, x, v, γ, fc)
    m12 = barra(NODOS, 5, 6, 0, 240, 150, 240, b_VIG, h_VIG, x, v, γ, fc)
    m13 = barra(NODOS, 6, 7, 150, 240, 300, 240, b_VIG, h_VIG, x, v, γ, fc)
    m14 = barra(NODOS, 7, 8, 300, 240, 450, 240, b_VIG, h_VIG, x, v, γ, fc)
elif test == 1: # testing
    m1 = barra(NODOS, 0, 1, 0, 0, 0, 100, b_COL, h_COL, x, v, γ, fc)
    m2 = barra(NODOS, 0, 2, 300, 0, 300, 100, b_COL, h_COL, x, v, γ, fc)
    m3 = barra(NODOS, 1, 2, 0, 100, 300, 100, b_VIG, h_VIG, x, v, γ, fc)

for i in range(1, struct.max_barras + 1):
    exec('barras.append(m' + str(i) + ')')
return barras

miembro = [None] + armar_estructura(TEST, struct.nodos)

print(miembro[1].a)

if 1:
    perfiles_i = [
        (58, 58), (58, 58), (58, 58), (58, 58), (54, 54), (54, 54), (54, 54), (54, 54), (49, 49), (49, 49), (49,
49), (49, 49), (49, 49), (49, 49), (49, 49), (29, 72), (29, 72), (29, 72), (39, 77), (39, 77), (39, 77), (29,
63), (29, 63), (29, 63), (29, 63), (29, 63), (29, 63),
    ]
    for i, m in enumerate(miembro):
        if not m: continue
        (m.b, m.h) = perfiles_i[i - 1]
        m.update()

# K del sistema
for m in miembro:
    if m == None: continue
    struct.K += m.T.T@m.K@m.T

def h_trib_losa(L, d, c_borde, neg=False): #actualizar descripcion
    """Retorna la proyeccion de los lados de un trapezio sobre su
    base y el valor del tramo constante de la carga distribuida 'q'.
    L = Largo de la base de la losa planta (viga del marco en
    analisis)
    d = Distancia entre marcos
    t = Espesor de la losa
    γ = Densidad del hormigon
    c_borde = Condicion de borde
    neg = Indica si las proyecciones 'a' y 'b' se deben voltear
    Por defecto se calcula siempre 'a' adyacente a un angulo de 45\
    grados cuando es posible."""
    if c_borde == 1: pass
    elif c_borde == '2':
        if d <= L:
            h = d/2**0.5*sin(60/180*π)/sin(75/180*π)
        else:
            h = L/2**0.5*sin(60/180*π)/sin(75/180*π)
        a = h
        b = a**2**0.5*sin(75/180*π)/sin(60/180*π) - a
    elif c_borde == '3c':
        k = 2/2**0.5*sin(60/180*π)/sin(75/180*π)
        if d <= k*L:
            h = d/2
        else:
            h = k*L/2
        a = h
        b = a**2**0.5*sin(75/180*π)/sin(60/180*π) - a
    elif c_borde == '3u':
        k = 2**0.5/2*sin(75/180*π)/sin(60/180*π)
        if d <= k*L:
            h = d/2**0.5*sin(60/180*π)/sin(75/180*π)
        else:
            h = L/2

```

```

    a = b = h
    elif c_borde == '4' or c_borde == '0':
        print('estoy aca')
        if d <= L:
            h = d/2
        else:
            h = L/2
            a = b = h
    if neg: a, b = b, a
    return [a, b, h]

if TEST: print(h_trib_losa(451.1, 355.77, '3c', 1))

def EP_rec(L, q, a, b, d):
    """Calculo de las reacciones en empotramiento perfecto para una carga constante.
    L = Largo del elemento
    q = Magnitud de la carga distribuida
    a = Distancia desde el inicio del elemento hasta la mitad de la aplicacion de la carga
    b = Distancia desde el final del elemento hasta la mitad de la aplicacion de la carga
    d = Largo de la aplicacion de la carga"""
    Ra = q*d/L**3*((2*a + L)*b**2 + (a - b)/4*d**2)
    Rb = q*d/L**3*((2*b + L)*a**2 - (a - b)/4*d**2)
    Ma = q*d/L**2*(a*b**2 + d**2/12*(L - 3*b))
    Mb = -q*d/L**2*(a**2*b + d**2/12*(L - 3*a))
    if struct.VERB: print('Ra Ma Rb Mb', round(Ra, 1), round(Ma, 1), round(Rb, 1), round(Mb, 1))
    return Ra, Ma, Rb, Mb

EP_rec(2, 1000, 0.5 + 0.6, 0.9, 1.2);

def EP_tri(L, q, a):
    """Calculo de las reacciones en empotramiento perfecto para una carga lineal.
    L = Largo del elemento
    q = Magnitud final de la carga distribuida lineal
    a = Distancia desde el inicio del elemento hasta el final de la aplicacion de la carga"""
    Ra = q*a/20*(10 - a**2/L**2*(15 - 8*a/L))
    Rb = q*a**3/(20*L**2)*(15 - 8*a/L)
    Ma = q*a**2/30*(10 - a/L*(15 - 6*a/L))
    Mb = -q*a**3/(20*L)*(5 - 4*a/L)
    if struct.VERB: print('Ra Ma Rb Mb', round(Ra, 1), round(Ma, 1), round(Rb, 1), round(Mb, 1))
    return Ra, Ma, Rb, Mb

EP_tri(2, 1000, 0.5);

# superposicion de casos de carga de una viga cualquiera respecto del eje Y

def emp_perf_Y(L, a, b, q):
    """Calcula las reacciones de empotramiento perfecto de una viga dada sometida a una carga trapezoidal.
    L = Largo del elemento
    a = Distancia desde el inicio del elemento hasta el inicio de la carga constante
    b = Distancia desde el final del elemento hasta el final de la carga constante
    q = Magnitud de la carga distribuida trapezoidal"""
    r = [[], [], []]
    joints = []
    d = L - a - b

    *r[0], = EP_tri(L, q, a)
    *r[1], = EP_rec(L, q, a + d/2, b + d/2, d)
    *r[2], = EP_tri(L, q, b)

    r[2][0], r[2][2] = r[2][2], r[2][0]
    r[2][1], r[2][3] = -r[2][3], -r[2][1]

    for i in range(4):
        joints.append(0)
        for reaction in r:
            joints[i] += reaction[i]

    if struct.VERB: # borrame
        line()
        print(
            'Va:', round(joints[0], 2),
            '\nMa:', round(joints[1], 2),
            '\nVb:', round(joints[2], 2),
            '\nMb:', round(joints[3], 2)
        )
    return (0, *joints[0:2], 0, *joints[2:4])

emp_perf_Y(3, 0.75, 1.4, 300)

# cargas muertas - peso propio

def pp_barras(accion='cargar'):
    """Aplica o limpia las cargas distribuidas y reacciones en los nodos producto del peso propio de todas las barras."""
    if accion == 'limpiar':
        for m in miembro:

```

```

        if not m: continue
        if not m.hayPP:
            print('no habia nada que eliminar!')
        else:
            for i, basicLoad in enumerate(m.dst_loads):
                if basicLoad[0] == '_pp-bar':
                    m.dst_loads.pop(i)
                    m.jnt_loads.pop(i)
                    break
            print('peso propio eliminado.')
            m.hayPP = False
    elif accion == 'cargar':
        for m in miembro:
            if not m: continue
            if m.hayPP: # actualizar PP para borrar el for
                print('ya estaba definido!')
            else:
                m.calcular_peso_propio()
                if m.name == 'VIGA':
                    for i, basicLoad in enumerate(m.dst_loads):
                        if basicLoad[0] == '_pp-bar':
                            m.jnt_loads.append((
                                '_pp-bar',
                                'D',
                                array(emp_perf_Y(
                                    m.L, *m.dst_loads[i][2]
                                ))
                            ))
                            break
                elif m.name == 'COL':
                    Na = m.dst_loads[0][2][2]*m.L/2
                    m.jnt_loads.append(
                        ('_pp-bar', 'D', array([0, Na, 0, 0, Na, 0]))
                    )
                print('peso propio calculado.')
                m.hayPP = True
    return

pp_barras('cargar')
#print(miembro[1].dst_loads)
#print(miembro[2].dst_loads)
#print(miembro[3].dst_loads)
#print(miembro[2].jnt_loads[0][2]*miembro[2].T

# cargas muertas y vivas - descarga losa

def q_losa(m, forma='', caso='D', accion='cargar', voltear=''): # m y returns temporales
    """Aplica o elimina las cargas distribuidas y las reacciones en los nodos a todas las vigas."""
    if accion == 'cargar':
        #for m in miembro:
        if not m: return #continue
        if m.name != 'VIGA': return #continue
        if caso != 'D' and caso != 'L':
            print('no trollees! que me crasheo.')
            return #break
        #if m.loso[caso][0] == 'existe': print(f'ya estaba cargado este caso ({caso}).')
        else:
            new_index = len(m.dst_loads)
            m.loso[caso].append(new_index)
            if voltear == 'mirror':
                q_losa = h_trib_losa(m.L, d_losas, forma, True) # [a, b, h]
            else:
                q_losa = h_trib_losa(m.L, d_losas, forma) # [a, b, h]
            q_losa[2] *= q_L if caso == 'L' else t_losas*m.y # [a, b, q]
            # rectificacion
            k = (2*m.L - q_losa[0] - q_losa[1])/(2*m.L)
            if struct.VERB:
                print('L', m.L, 'a', q_losa[0], 'b', q_losa[1])
                print('k', k)
                print('q_losa antes', q_losa)
            q_losa = [0, 0, k*q_losa[2]]
            if struct.VERB: print('q_losa despues', q_losa)
            m.dst_loads.append((
                '_q-loso',
                caso,
                tuple(q_losa)
            ))
            m.jnt_loads.append((
                '_q-loso',
                caso,
                array( emp_perf_Y(m.L, *m.dst_loads[m.loso[caso][-1]][2] )
            ))
            print(f'se ha cargado la losa (caso: {caso}) en el elemento {m.name}-i:{m.i}.')
            line()
            line()
            #m.loso[caso][0] = 'existe'

```



```

elif accion == 'l': # limpiar... pero algun dia
    print('nada que ver aquí :v *se crashea*')
    raise
return

if TEST == 3:
    if 1: # !! no sobrescribe
        q_losa(miembro[17], '2', 'D', 'cargar', 'mirror') # piso 1
        q_losa(miembro[17], '3c', 'D', 'cargar', 'mirror')
        q_losa(miembro[18], '3u', 'D')
        q_losa(miembro[18], '3u', 'D')
        q_losa(miembro[19], '2', 'D')
        q_losa(miembro[19], '3c', 'D')
        q_losa(miembro[17], '2', 'L', 'cargar', 'mirror')
        q_losa(miembro[17], '3c', 'L', 'cargar', 'mirror')
        q_losa(miembro[18], '3u', 'L')
        q_losa(miembro[18], '3u', 'L')
        q_losa(miembro[19], '2', 'L')
        q_losa(miembro[19], '3c', 'L')
        q_losa(miembro[20], '2', 'D', 'cargar', 'mirror') # piso 2
        q_losa(miembro[20], '3c', 'D', 'cargar', 'mirror')
        q_losa(miembro[21], '3u', 'D')
        q_losa(miembro[21], '3u', 'D')
        q_losa(miembro[22], '2', 'D')
        q_losa(miembro[22], '3c', 'D')
        q_losa(miembro[20], '2', 'L', 'cargar', 'mirror')
        q_losa(miembro[20], '3c', 'L', 'cargar', 'mirror')
        q_losa(miembro[21], '3u', 'L')
        q_losa(miembro[21], '3u', 'L')
        q_losa(miembro[22], '2', 'L')
        q_losa(miembro[22], '3c', 'L')
        q_losa(miembro[23], '2', 'D', 'cargar', 'mirror') # piso 3
        q_losa(miembro[23], '3c', 'D', 'cargar', 'mirror')
        q_losa(miembro[24], '3u', 'D')
        q_losa(miembro[24], '3u', 'D')
        q_losa(miembro[25], '2', 'D')
        q_losa(miembro[25], '3c', 'D')
        q_losa(miembro[23], '2', 'L', 'cargar', 'mirror')
        q_losa(miembro[23], '3c', 'L', 'cargar', 'mirror')
        q_losa(miembro[24], '3u', 'L')
        q_losa(miembro[24], '3u', 'L')
        q_losa(miembro[25], '2', 'L')
        q_losa(miembro[25], '3c', 'L')
        q_losa(miembro[26], '2', 'D', 'cargar', 'mirror') # piso 4
        q_losa(miembro[26], '3c', 'D', 'cargar', 'mirror')
        q_losa(miembro[27], '3u', 'D')
        q_losa(miembro[27], '3u', 'D')
        q_losa(miembro[28], '2', 'D')
        q_losa(miembro[28], '3c', 'D')
        q_losa(miembro[26], '2', 'L', 'cargar', 'mirror')
        q_losa(miembro[26], '3c', 'L', 'cargar', 'mirror')
        q_losa(miembro[27], '3u', 'L')
        q_losa(miembro[27], '3u', 'L')
        q_losa(miembro[28], '2', 'L')
        q_losa(miembro[28], '3c', 'L')
        [print(m.dst_loads) for m in miembro if m];
        [print(m.jnt_loads) for m in miembro if m];
    elif TEST == 1:
        if 1: # !! no sobrescribe
            q_losa(miembro[3], '2', 'D')
            q_losa(miembro[3], '3c', 'D')
            q_losa(miembro[3], '2', 'L')
            q_losa(miembro[3], '3c', 'L')
            [print(m.dst_loads) for m in miembro if m];
            [print(m.jnt_loads) for m in miembro if m];

# cargas sismicas - metodo estatico
def metodo_estatico(zona, categoria, suelo, R, h_pisos, n_pisos, d_losas, n_vanos):
    """Calcula un vector con las fuerzas horizontales producto del sismo."""
    TABLA_I = { # Tabla 6.1
        # "CategoriaDeEdificio": I [factor de importancia],
        "I": 0.6,
        "II": 1.0,
        "III": 1.2,
        "IV": 1.2,
    }

    TABLA_A0 = { # Tabla 6.2
        # "ZonaSismica": A0/g,
        "1": 0.20,
        "2": 0.30,
        "3": 0.40,
    }

    TABLA_PARAMETROS_SUELO = { # Tabla 6.3
        # "TipoDeSuelo": (S, T0, Tp [T'], n, p),

```

```

    "A": (0.90, 0.15, 0.20, 1.00, 2.0),
    "B": (1.00, 0.30, 0.35, 1.33, 1.5),
    "C": (1.05, 0.40, 0.45, 1.40, 1.6),
    "D": (1.20, 0.75, 0.85, 1.80, 1.0),
    "E": (1.30, 1.20, 1.35, 1.80, 1.0),
    "F": (None, None, None, None, None),
}

TABLA_C_MAX = { # Tabla 6.4
    # "R": CoeficienteC_max,
    "2": 0.90,
    "3": 0.60,
    "4": 0.55,
    "5.5": 0.40,
    "6": 0.35,
    "7": 0.35,
}

Tn = round(0.09*h_pisos*n_pisos/(d_losas/100*n_vanos)**0.5, 4)

A0 = TABLA_A0[zona] # aceleracion efectiva
I = TABLA_I[categoria]
S, T0, Tp, n, p = TABLA_PARAMETROS_SUELO[suelo]

# coeficiente sismico
Cmax = round(TABLA_C_MAX[str(R)]*S*A0, 8)
Cmin = A0*S/6
C = 2.75*S*A0/R*(Tp/Tn)**n

if C < Cmin: C = Cmin
elif C > Cmax: C = Cmax

# numero de elementos por piso
n_viga = n_marcos*n_vanos
n_colum = n_marcos*(n_vanos + 1)
n_vig_u = (n_marcos - 1)*(n_vanos + 1) # viga de union de marcos

# masa por elemento
for m in miembro:
    if not m: continue
    if m.name == 'VIGA':
        m_viga = m.y*m.A*m.L
        print('viga: gamma, area, largo', m.y, m.A, m.L)
        m_vig_u = m.y*m.A*d_losas
        break
for m in miembro:
    if not m: continue
    if m.name == 'COL':
        m_colum = m.y*m.A*m.L
        print('col: gamma, area, largo', m.y, m.A, m.L)
        break

if struct.VERB:
    print('numero de v, v_u, c:', n_viga, n_vig_u, n_colum)
    print('masa de v, v_u, c:', m_viga, m_vig_u, m_colum)

# masa estructura por piso
M_st = (n_viga*m_viga) + (n_colum*m_colum) + (n_vig_u*m_vig_u)
print(f'n y m: viga {n_viga} de {m_viga}, col {n_colum} de {m_colum}, viga_un {n_vig_u} de {m_vig_u}')
# masa losas por piso
M_l = m.y * b_losas*d_losas*t_losas * n_vanos*(n_marcos - 1)

# masa sismica por nivel
MUERTO = M_st + M_l # [kg]
VIVO = 0.25*Q_L * b_losas*n_vanos * d_losas*(n_marcos - 1) # [kg]
Pk = [0]
for nivel in range(n_pisos):
    if nivel == n_pisos - 1:
        Pk.append(MUERTO - n_colum*m_colum/2 + VIVO)
        break
    Pk.append(MUERTO + VIVO)
print('masas por nivel:', Pk, 'largo pk:', len(Pk))

P = sum(Pk) # [kg] - peso sismico
Q0 = C*I*P # [kg] - corte basal

print(
    'testing vars:\n',
    f'A0: {A0}, I: {I}, S: {S}, T0: {T0}\n',
    f'Tp: {Tp}, n: {n}, p: {p}\n',
    f'Cmax: {Cmax}, Cmin: {round(Cmin, 4)}, C: {round(C, 4)}\n',
    f'Q0: {Q0}, Pk: {Pk}, P: {P}\n',
    f'nperiodo fundamental: {Tn}\n'
) # borrame

```

```

print('masas: M_st {}, M losa {}'.format(M_st, M_l))
print('muerto y vivo:', MUERTO, VIVO)

H = round(h_pisos*n_pisos, 4)
Z = [0] + [round(i*h_pisos, 4) for i in range(1, n_pisos + 1)]
A = [0]
for k, _ in enumerate(Z):
    if k == 0: continue
    A.append(
        (1 - Z[k - 1]/H)**0.5 - (1 - Z[k]/H)**0.5
    )
    print('nivel {}, altura acumulada {}, {}'.format(k, Z[k], round(A[k], 3))) # borrame

P = Pk # mejor coincidencia con formulas de la nch433
del Pk

print('largos:', len(A), len(P))
F = []
AjPj = sum([i*k for i, k in zip(A, P)]) #  $\sum A_j * P_j$ 
for k, _ in enumerate(P):
    F.append(A[k]*P[k]/AjPj*Q0)
    print('AP/AP:', A[k]*P[k]/AjPj)

print('A*P', AjPj) # borrame
print('F por nivel:', F) # borrame

fx = []
for k, _ in enumerate(F):
    if k == 0: continue
    for i in range(n_vanos + 1):
        fx.append(F[k]/((n_vanos + 1)*(n_marcos)))
        fx.append(0)
        fx.append(0)
return array(fx).T

E = metodo_estatico(zona, categoria, suelo, R, h_pisos/100, n_pisos, d_losas, n_vanos)
E

# vector de fuerzas segun casos de carga
# 1.4*D
# 1.2*D + 1.6*L
# 1.2*D + 1.4*E + L
# 1.2*D - 1.4*E + L
# 0.9*D + 1.4*E
# 0.9*D - 1.4*E

def loads_gen(accion='generar'): # convertir en metodo
    """Genera las categorías y combinaciones de carga de la estructura."""
    if accion == 'generar':
        case_gen = ['D', 'L', 'E']
        struct.caso = {}
        for i in case_gen:
            struct.caso.update({ i: zeros(len(struct.K)) })

        combo_gen = [
            '1.4 D',
            '1.2 D + 1.6 L',
            '1.2 D + L',
            'E',
            '- E',
            '1.2 D + 1.4 E + L',
            '1.2 D - 1.4 E + L',
            '0.9 D + 1.4 E',
            '0.9 D - 1.4 E',
        ]
        struct.combo = {}
        struct.r = {}
        for m in miembro:
            if not m: continue
            if 1: m.q_combo.clear() # temp carga rectangular
            m.combo.clear()
            for i in combo_gen:
                struct.combo.update({ i: zeros(len(struct.K)) })
                struct.r.update({ i: zeros(len(struct.K)) })
                for m in miembro:
                    if not m or i == 'E' or i == '- E': continue
                    if struct.VERB: print(f'limpiando y generando {m.name}')
                    if 1: m.q_combo.update({ i: 0 }) # temp carga rectangular
                    m.combo.update({ i: zeros(len(struct.K)) })
            elif accion == 'cargar': pass
        return

loads_gen()

for i, m in enumerate(miembro):
    if not m: continue

```

```

m.case_gen()
for F in m.jnt_loads:
    if 0: # borrame
        print(
            f'Fuente: miembro {i}',
            f'perfil: {m.name}',
            f'nombre: \'{F[0]}\'',
            f'caso: \'{F[1]}\'',
            f'vector: {F[2]}\n',
            sep='\n'
        )
        if F[1] == 'D':
            m.caso['D'] += F[2]
            struct.caso['D'] -= F[2]@m.T
        elif F[1] == 'L':
            m.caso['L'] += F[2]
            struct.caso['L'] -= F[2]@m.T
if 1: # temp rect
    for F in m.dst_loads:
        if F[1] == 'D':
            m.q_caso['D'] += F[2][2]
        if F[1] == 'L':
            m.q_caso['L'] += F[2][2]

if TEST != 2:
    struct.caso['E'] = metodo_estatico(zona, categoria, suelo, R, h_pisos/100, n_pisos, d_losas, n_vanos)

if 0: # nani
    for m in miembro:
        if m: print(f'\nperfil \'{m.name}\'', m.caso, sep='\n')

if 1: # temp
    for m in miembro:
        if not m: continue
        # cargas rectangulares por combinacion
        m.q_combo['1.4 D'] = \
            1.4*m.q_caso['D']

        m.q_combo['1.2 D + 1.6 L'] = \
            1.2*m.q_caso['D'] + 1.6*m.q_caso['L']

        m.q_combo['1.2 D + L'] = \
            1.2*m.q_caso['D'] + m.q_caso['L']

        m.q_combo['1.2 D + 1.4 E + L'] = \
            1.2*m.q_caso['D'] + m.q_caso['L']

        m.q_combo['1.2 D - 1.4 E + L'] = \
            1.2*m.q_caso['D'] + m.q_caso['L']

        m.q_combo['0.9 D + 1.4 E'] = \
            0.9*m.q_caso['D']

        m.q_combo['0.9 D - 1.4 E'] = \
            0.9*m.q_caso['D']

for m in miembro:
    if not m: continue
    # vectores de empotramiento perfecto por combinacion
    m.combo['1.4 D'] = \
        1.4*m.caso['D']

    m.combo['1.2 D + 1.6 L'] = \
        1.2*m.caso['D'] + 1.6*m.caso['L']

    m.combo['1.2 D + L'] = \
        1.2*m.caso['D'] + m.caso['L']

    m.combo['1.2 D + 1.4 E + L'] = \
        1.2*m.caso['D'] + m.caso['L']

    m.combo['1.2 D - 1.4 E + L'] = \
        1.2*m.caso['D'] + m.caso['L']

    m.combo['0.9 D + 1.4 E'] = \
        0.9*m.caso['D']

    m.combo['0.9 D - 1.4 E'] = \
        0.9*m.caso['D']

# vectores de fuerza por combinacion
struct.combo['1.4 D'] = \
    1.4*struct.caso['D']

struct.combo['1.2 D + 1.6 L'] = \
    1.2*struct.caso['D'] + 1.6*struct.caso['L']

```

```

struct.combo['1.2 D + L'] = \
    1.2*struct.caso['D'] + struct.caso['L']

struct.combo['E'] = \
    struct.caso['E']

struct.combo['- E'] = \
    -1*struct.caso['E']

struct.combo['1.2 D + 1.4 E + L'] = \
    1.2*struct.caso['D'] + 1.4*struct.caso['E'] + struct.caso['L']

struct.combo['1.2 D - 1.4 E + L'] = \
    1.2*struct.caso['D'] - 1.4*struct.caso['E'] + struct.caso['L']

struct.combo['0.9 D + 1.4 E'] = \
    0.9*struct.caso['D'] + 1.4*struct.caso['E']

struct.combo['0.9 D - 1.4 E'] = \
    0.9*struct.caso['D'] - 1.4*struct.caso['E']

vars(struct)

line()
struct.r = {}
for COMBO in struct.combo.keys():
    struct.r.update({ COMBO: inv(struct.K)@struct.combo[COMBO] }) # desplazamientos

for m in miembro:
    if not m: continue
    # inicializacion
    m.r = {}
    m.e = {}
    m.s = {}
    m.v = {}
    m.sigma = {}
    for COMBO in m.combo.keys():
        # definicion
        m.r.update({ COMBO: m.T@struct.r[COMBO] })
        m.e.update({ COMBO: m.a@m.r[COMBO] })
        m.s.update({ COMBO: m.k@m.e[COMBO] })
        m.v.update({ COMBO: (m.s[COMBO][0] + m.s[COMBO][1])/m.L })
        m.sigma.update({ COMBO: m.a.T@m.s[COMBO] })
    for i, m in enumerate(miembro):
        if m and struct.VERB: print(f'miembro {i}, vector de esfuerzos:\n', m.sigma)
#vars(miembro[1])

for m in miembro:
    if not m: continue
    m.resumen = {}
    for COMBO in m.sigma.keys():
        m.resumen.update({ COMBO: mround(m.sigma[COMBO] + m.combo[COMBO], 2) })

def reporte(resumen='esfuerzos'):
    """Genera los reportes para esfuerzos en los extremos de cada
    elemnto, desplazamientos nodales, desplazamientos por nivel y
    drift por nivel para cada combinacion y les da formato en un
    archivo por cada tabla."""
    if resumen != 'generar':
        tabla = struct.annainfo[resumen]
    if resumen == 'generar':
        struct.clean()
        tabla = struct.annainfo['esfuerzos']
        for k, m in enumerate(miembro):
            if not m: continue
            for COMBO in m.resumen:
                if m.name == 'VIGA':
                    tabla['body'].append([
                        m.name,
                        k,
                        COMBO,
                        'i',
                        m.resumen[COMBO][0],
                        m.resumen[COMBO][1],
                        m.resumen[COMBO][2]/100,
                    ])
                elif m.name == 'COL':
                    tabla['body'].append([
                        m.name,
                        k,
                        COMBO,
                        'j',
                        m.resumen[COMBO][3]*-1,
                        m.resumen[COMBO][4]*-1,
                        m.resumen[COMBO][5]/-100,
                    ])
            elif m.name == 'COL':
                tabla['body'].append([
                    m.name,
                    k,
                    COMBO,
                    'j',
                    m.resumen[COMBO][3]*-1,
                    m.resumen[COMBO][4]*-1,
                    m.resumen[COMBO][5]/-100,
                ])

```

```

        k,
        COMBO,
        'j',
        m.resumen[COMBO][1],
        m.resumen[COMBO][0]*-1,
        m.resumen[COMBO][2]/100,
    ])
    tabla['body'].append([
        'j',
        m.resumen[COMBO][4]*-1,
        m.resumen[COMBO][3],
        m.resumen[COMBO][5]/-100,
    ])
    tabla = struct.annainfo['delta-nodos']
    for combo in struct.r:
        for nodo in range(struct.nodos):
            tabla['body'].append([
                combo,
                nodo + 1,
                struct.r[combo][3*nodo]*10,
                struct.r[combo][3*nodo + 1]*10,
                struct.r[combo][3*nodo + 2],
            ])
    tabla = struct.annainfo['drifts']
    for combo in ['E', '- E']:
        struct.drift.update({ combo: [] })
        δmax1, δmin1, δmax2, δmin2 = 0, 0, 0, 0
        nivel = 0
        deltax = []
        for i in range(len(struct.r[combo])):
            if 3*i >= len(struct.r[combo]): break
            deltax.append(struct.r[combo][3*i])
        for i in range(len(deltax)):
            if struct.dim['x']*i >= len(deltax): break
            nivel += 1
            for j in deltax[struct.dim['x']*i:struct.dim['x']*(i + 1)]:
                if deltax[struct.dim['x']*i] == j:
                    δmin2 = j
                    if abs(j) > abs(δmax2):
                        δmax2 = j
                    elif abs(j) < abs(δmin2):
                        δmin2 = j
            drift = round((δmax2 - δmin1)/h_pisos*1000, 3) # [%]
            δmax1, δmin1 = δmax2, δmin2
            tabla['body'].append([
                combo,
                nivel,
                10*δmax2,
                10*δmin2,
                drift,
            ])
        struct.drift[combo].append(drift)
    elif resumen == 'esfuerzos':
        print(' '*30 + '{:>19}{:>13}{:>13}'.format(*tabla['head'][1]))
        print('fila  {:^6}  {:^19}  {:^19}  {:^19}{:>12}{:>13}{:>13}'.format(*tabla['head'][0]))
        i = 1
        print('='*75)
        for fila in tabla['body']:
            #if i == 12: break
            if i%2 != 0:
                print(f'{i:<6}' + '{:<4}{:>2}  {:^19}  ({} )  {:>12}  {:>12}  {:>12.2f}'.format(*fila))
            else:
                print(f'{i:<6}' + ' '*27 + '({})  {:>12}  {:>12}  {:>12.2f}'.format(*fila))
                print('-'*75)
            i += 1
    elif resumen == 'delta-nodos':
        print(' '*22 + '{:>21}{:>12}{:>14}'.format(*tabla['head'][1]))
        print('fila  {:^19}  {:^4}  {:>12}  {:>11}  {:>13}'.format(*tabla['head'][0]))
        i = 1
        print('='*69)
        for fila in tabla['body']:
            #if i == 12: break
            print(f'{i:<6}' + '{:^19}  ({}:02)  {:>12.4f}  {:>11.4f}  {:>13.3e}'.format(*fila))
            print('-'*69)
            i += 1
    elif resumen == 'drifts':
        print(' '*22 + '{:>21}{:>12}{:>11}'.format(*tabla['head'][1]))
        print('fila  {:^19}  {:^4}  {:>11}  {:>11}  {:>10}'.format(*tabla['head'][0]))
        i = 1
        print('='*66)
        for fila in tabla['body']:
            #if i == 12: break
            print(f'{i:<6}' + '{:^19}  ({}:02)  {:>12.4f}  {:>11.4f}  {:>10.3f}'.format(*fila))
            print('-'*66)
            i += 1
    return

```

```

reporte('generar')
#reporte('esfuerzos')
#reporte('delta-nodos')
reporte('drifts')

ψ = 0
d = 0
for i in struct.drift:
    for j in struct.drift[i]:
        if abs(j) > d:
            d = abs(j)
            r = abs(j)/(2/1.2)
ψ = r**0.25

p(f'drift: {d}, r: {r}, ψ: {ψ}')

p(f'drift maximo: {d}\n',
  f'ψ: {ψ}', sep='')
line()

if 1:
    perfiles_i = [(m.b, m.h) for m in miembro if m]
    perfiles_f = [(round(m.b*ψ), round(m.h*ψ)) for m in miembro if m]
    if perfiles_i == perfiles_f: p('verifica para el drift!')
    else: p('se necesita corregir los perfiles...')
    line()
    for m in miembro:
        if not m: continue
        p(m.b, m.h, m.name)
    line()
    for m in miembro:
        if not m: continue
        p(
            (round(m.b*ψ), round(m.h*ψ)), ', ',
            sep=',', end='')
    p()
    line()

def porto():
    _perfiles = []
    for m in miembro:
        if not m: continue
        _perfiles.append((m.b, m.h))
    dick = struct.annainfo['esfuerzos'].copy()
    dick.update({'perfiles': _perfiles })
    dick.update({'drift': d })
    with open('_pera.py', 'w') as _pera:
        _pera.write('tabla = ' + str(dick))

porto()
del porto

def N(x, L, a, b, q, Na, Va, Ma):
    """Calcula el diagrama de esfuerzo axial de una viga."""
    if 0 <= x <= L:
        return Na
    else:
        return 'out of bar length!'

def V(x, L, a, b, q, Na, Va, Ma): # sera metodo
    """Calcula el diagrama de esfuerzo cortante de una viga."""
    def V1(x): return Va - q*x**2/(2*a)
    def V2(x): return V1(a) - q*(x - a)
    def V3(x): return V2(L - b) - q*(x - L + b)/2*(1 + (L - x)/b)

    if 0 <= x < a: return V1(x)
    elif a <= x < L - b: return V2(x)
    elif L - b <= x <= L: return V3(x)

    return 'out of bar length!'

def M(x, L, a, b, q, Na, Va, Ma):
    """Calcula el diagrama de momento de una viga."""
    def M1(x): return -Ma + Va*x - q*x**3/(6*a)
    def M2(x): return -Ma + Va*x - q*(a*x/2 - a**2/3) - q*(x - a)**2/2
    def M3(x): return -Ma + Va*x - q*(a*x/2 - a**2/3) - q*(L - b - a)*(x - a - (L - b - a)/2) - q*(x - L
+ b)**2/3*(1 + (L - x)/(2*b))

    if 0 <= x < a: return M1(x)
    elif a <= x < L - b: return M2(x)
    elif L - b <= x <= L: return M3(x)

    return 'out of bar length!'

```

```

def grafVV(barras, barra, combo, ij=False):
    """Grafica los diagramas de esfuerzo interno de cada elemento."""

    # q = ( a, b, q )
    q_rect = (
        0.00000001,
        0.00000001,
        barras[barra].q_combo[combo],
    )

    Na = barras[barra].resumen[combo][0]
    Va = barras[barra].resumen[combo][1]
    Ma = barras[barra].resumen[combo][2]
    L = int(barras[barra].L)
    x, y = [], {'N': [], 'V': [], 'M': []}
    ijab = {'N': [], 'V': [], 'M': []}

    if barras[barra].name == 'VIGA' and ij:
        for m in barras:
            if not m: continue
            if m.j == barras[barra].i and m.name == 'COL':
                L1 = m.b/2
                L2 = L - m.b/2
                ijab['N'].append( N(L1, L, *q_rect, Na, Va, Ma) )
                ijab['V'].append( V(L1, L, *q_rect, Na, Va, Ma) )
                ijab['M'].append( M(L1, L, *q_rect, Na, Va, Ma)/100 )
                ijab['N'].append( N(L2, L, *q_rect, Na, Va, Ma) )
                ijab['V'].append( V(L2, L, *q_rect, Na, Va, Ma) )
                ijab['M'].append( M(L2, L, *q_rect, Na, Va, Ma)/100 )
            print(
                'Extremo i:',
                f'\nN({L1}):', round( ijab['N'][0], 2 ), '[kg]'
                f'\nV({L1}):', round( ijab['V'][0], 2 ), '[kg]'
                f'\nM({L1}):', round( ijab['M'][0], 2 ), '[kg-m]'
            )
            print(
                'Extremo j:',
                f'\nN({L2}):', round( ijab['N'][1], 2 ), '[kg]'
                f'\nV({L2}):', round( ijab['V'][1], 2 ), '[kg]'
                f'\nM({L2}):', round( ijab['M'][1], 2 ), '[kg-m]'
            )
        )

    for _x in range(L + 1):
        x.append(_x)
        y['N'].append(N(_x, L, *q_rect, Na, Va, Ma))
        y['V'].append(V(_x, L, *q_rect, Na, Va, Ma))
    if barras[barra].name == 'VIGA':
        for _x in range(L + 1):
            y['M'].append(M(_x, L, *q_rect, Na, Va, Ma))
    elif barras[barra].name == 'COL':
        for _x in range(L + 1):
            y['M'].append(M(_x, L, *q_rect[0:2], 0, Va, Na*-1, Ma))
    palo = [0 for i in x]
    x, palo = array(x), array(palo)
    if barras[barra].name == 'VIGA':
        y['N'] = array(y['N'])
        y['V'] = array(y['V'])
        y['M'] = array(y['M'])/100
    elif barras[barra].name == 'COL':
        y['N'], y['V'] = array(y['V']), array(y['N'])*-1
        y['M'] = array(y['M'])/100

    figura, ejes = pete.subplots(3, 1, figsize=(8.0,11.0))
    pete.rcParams.update({'font.size': 8.7})
    pete.subplots_adjust(hspace=0.4)
    ejes[0].set_title(f'Miembro {barra}: {barras[barra].name}', y=1.1, fontsize=19)
    for k, nvm in enumerate(['N', 'V', 'M']):
        ejes[k].grid()
        ejes[k].plot(x, y[nvm], 'k--', label=f'${nvm}(x)$')
        if barras[barra].name == 'VIGA':
            ejes[k].plot(L1, ijab[nvm][0], 'ro')
            ejes[k].plot(L2, ijab[nvm][1], 'ro')
            if 0 < min(y[nvm]) <= max(y[nvm]):
                ejes[k].plot([L1, L1], [0, max(y[nvm])], 'r', linewidth=1)
                ejes[k].plot([L2, L2], [0, max(y[nvm])], 'r', linewidth=1)
            elif min(y[nvm]) <= max(y[nvm]) < 0:
                ejes[k].plot([L1, L1], [min(y[nvm]), 0], 'r', linewidth=1)
                ejes[k].plot([L2, L2], [min(y[nvm]), 0], 'r', linewidth=1)
            else:
                ejes[k].plot([L1, L1], [min(y[nvm]), max(y[nvm])], 'r', linewidth=1)
                ejes[k].plot([L2, L2], [min(y[nvm]), max(y[nvm])], 'r', linewidth=1)
        ejes[k].plot(x, palo, 'k-')
        ejes[k].fill_between(x, y[nvm], palo, where=(y[nvm] > palo), facecolor='b', alpha=0.33)
        ejes[k].fill_between(x, y[nvm], palo, where=(y[nvm] < palo), facecolor='g', alpha=0.33)
        ejes[k].legend(loc='best', fontsize=15)
        ejes[k].set_xlabel(f'$L(x)$ [cm]', fontsize=14)
        if nvm == 'M':

```



```

        ejes[k].set_ylabel(f'${nvm}(x)$ [kg-m]', fontsize=14)
        ejes[k].invert_yaxis()
    else:
        ejes[k].set_ylabel(f'${nvm}(x)$ [kg]', fontsize=14)
    #figura.show()
    figura.savefig(f'_miembro{barra}.svg')
    return

if 0:
    if TEST == 3:
        grafVV(miembro, 18, '1.2 D + 1.4 E + L', 1)
    elif TEST == 1:
        grafVV(miembro, 3, '0.9 D - 1.4 E', 1)

#fMaxViga = [CorteA, MomentoA, CorteB, MomentoB]
#fMaxCol = [AxialA, CorteA, MomentoA, AxialB, CorteB, MomentoB]
#perfiles_0 = (anchoV, altoV, anchoC, altoC)
#perfiles_1 = (anchoV, altoV, anchoC, altoC)
#import lo_del_ale as optijandro

def pAPP():
    leerInput()
    armarEstructura()
    optimus = False
    while not optimus:
        cargasBasicas()
        cargarCombos()
        perfiles_0, esfuerzos = anna()
        perfiles_1, salida = optijandro(perfiles_0, esfuerzosV)
        perfiles_1, salida = optijandro(perfiles_0, esfuerzosC)
        if perfiles_0 == perfiles_1:
            optimus = True
        else:
            perfiles_0 = perfiles_1
            perfiles_1 = (None,)*4
            borrarFuerzas()
    reporte(salida)
    return 0

#ned

if MERYTEST:
    print(
        f'Current memory: {tracemalloc.get_traced_memory()[0]/10**6:.2f} [MB]\n',
        f'Peak           : {tracemalloc.get_traced_memory()[1]/10**6:.2f} [MB]',
        sep=''
    )
    #tracemalloc.stop()
varsC(dir())
#vars(struct)

import matplotlib.pyplot as plt
import _pera as p

def optijandro():
    tabla=p.tabla['body']

    def filtroCV(combis, combi_e, combi_s, tab, largosV, largosC):
        bars1=[[tab[i-1]+tab[i] for j in range(2) if j==1]
                for i in range(len(tab)) if i%2!=0]
        bars2=[bars1[i][0] for i in range(len(bars1))]
        bars=[bars2[i] for i in range(combis*j, combis*j+combis)
              for j in range(int(len(bars2)/combis))]

        exc=[]
        for i in range(len(bars)):
            temp1=[]
            maxim=0
            for j in range(len(bars[0])):
                for k in range(len(bars[0][0])):
                    if bars[i][j][4]/bars[i][j][6]>bars[i][j][8]/bars[i][j][10]:
                        pu=bars[i][j][4]
                        mu=bars[i][j][6]
                    else:
                        pu=bars[i][j][8]
                        mu=bars[i][j][10]
                    ex=mu/pu
                    if ex>maxim:
                        maxim=ex
                        list1=[round(mu,1), round(pu,1), round(ex,3)]
            temp1.append(list1)
            exc.append(temp1)

        bars_e1=[bars2[i] for i in range(len(bars2)) if 'E' in bars2[i][2]]

```

```

bars_s1=[bars2[i] for i in range(len(bars2)) if 'E' not in bars2[i][2]]
bars_e=[[bars_e1[i] for i in range(combi_e*j, combi_e*j+combi_e)]
        for j in range(int(len(bars_e1)/combi_e)]]
bars_s=[[bars_s1[i] for i in range(combi_s*j, combi_s*j+combi_s)]
        for j in range(int(len(bars_s1)/combi_s)]]
col_e=[[bars_e[j][i] for i in range(0, combi_e)] for j in range(len(bars_e))
        if bars_e[j][0][0]=='COL']
col_s=[[bars_s[j][i] for i in range(0, combi_s) if bars_s[j][i][2]!='1.2 D + L']
        for j in range(len(bars_s)) if bars_s[j][0][0]=='COL']

col_dl=[[bars_s[j][i] for i in range(0, combi_s) if bars_s[j][i][2]=='1.2 D + L']
        for j in range(len(bars_s)) if bars_s[j][0][0]=='COL']
vig_e=[[bars_e[j][i] for i in range(0, combi_e)]
        for j in range(len(bars_e)) if bars_e[j][0][0]=='VIGA']
vig_s=[[bars_s[j][i] for i in range(0, combi_s) if bars_s[j][i][2]!='1.2 D + L']
        for j in range(len(bars_s)) if bars_s[j][0][0]=='VIGA']
vig_dl=[[bars_s[j][i] for i in range(0, combi_s) if bars_s[j][i][2]=='1.2 D + L']
        for j in range(len(bars_s)) if bars_s[j][0][0]=='VIGA']

maTriX_ij = lambda lista: [[round(lista[k][j][i],1) for j in range(len(lista[0]))]
                             for i in [5,9]] for k in range(len(lista))]
maxTriX_i = lambda lista: [[round(max([lista[k][j][i] for j in range(len(lista[0]))]),2)
                             for i in [4,5,6]] for k in range(len(lista))]
minTriX_i = lambda lista: [[round(min([lista[k][j][i] for j in range(len(lista[0]))]),2)
                             for i in [4,5,6]] for k in range(len(lista))]
maxTriX_j = lambda lista: [[round(max([lista[k][j][i] for j in range(len(lista[0]))]),2)
                             for i in [8,9,10]] for k in range(len(lista))]
minTriX_j = lambda lista: [[round(min([lista[k][j][i] for j in range(len(lista[0]))]),2)
                             for i in [8,9,10]] for k in range(len(lista))]

npisos, nbahias = len(col_e)-len(vig_e), int(len(vig_e)/(len(col_e)-len(vig_e)))

forma_col = lambda lista, nbahias, npisos: [
    [lista[j] for j in range(i*(nbahias+1), (i+1)*(nbahias+1))] for i in range(npisos)]
forma_vig = lambda lista, nbahias, npisos: [
    [lista[j] for j in range(i*(nbahias), (i+1)*(nbahias))] for i in range(npisos)]

exc = forma_col(exc[:len(col_e)],nbahias, npisos)

max_col_ei = forma_col(maxTriX_i(col_e),nbahias,npisos)
max_col_si = forma_col(maxTriX_i(col_s),nbahias,npisos)
max_col_dli = forma_col(maxTriX_i(col_dl),nbahias,npisos)

min_col_ei = forma_col(minTriX_i(col_e),nbahias,npisos)
min_col_si = forma_col(minTriX_i(col_s),nbahias,npisos)
min_col_dli = forma_col(minTriX_i(col_dl),nbahias,npisos)

max_col_ej = forma_col(maxTriX_j(col_e),nbahias,npisos)
max_col_sj = forma_col(maxTriX_j(col_s),nbahias,npisos)
max_col_dlj = forma_col(maxTriX_j(col_dl),nbahias,npisos)

min_col_ej = forma_col(minTriX_j(col_e),nbahias,npisos)
min_col_sj = forma_col(minTriX_j(col_s),nbahias,npisos)
min_col_dlj = forma_col(minTriX_j(col_dl),nbahias,npisos)

mat_col_e = forma_col(maTriX_ij(col_e),nbahias,npisos)
mat_col_s = forma_col(maTriX_ij(col_s),nbahias,npisos)

max_vig_ei = forma_vig(maxTriX_i(vig_e),nbahias,npisos)
max_vig_si = forma_vig(maxTriX_i(vig_s),nbahias,npisos)
max_vig_dli = forma_vig(maxTriX_i(vig_dl),nbahias,npisos)

min_vig_ei = forma_vig(minTriX_i(vig_e),nbahias,npisos)
min_vig_si = forma_vig(minTriX_i(vig_s),nbahias,npisos)
min_vig_dli = forma_vig(minTriX_i(vig_dl),nbahias,npisos)

max_vig_ej = forma_vig(maxTriX_j(vig_e),nbahias,npisos)
max_vig_sj = forma_vig(maxTriX_j(vig_s),nbahias,npisos)
max_vig_dlj = forma_vig(maxTriX_j(vig_dl),nbahias,npisos)

min_vig_ej = forma_vig(minTriX_j(vig_e),nbahias,npisos)
min_vig_sj = forma_vig(minTriX_j(vig_s),nbahias,npisos)
min_vig_dlj = forma_vig(minTriX_j(vig_dl),nbahias,npisos)

mat_vig_e = forma_vig(maTriX_ij(vig_e),nbahias,npisos)
mat_vig_s = forma_vig(maTriX_ij(vig_s),nbahias,npisos)

matCorte_col=[mat_col_e,mat_col_s]
matCorte_vig=[mat_vig_e,mat_vig_s]

# 'axial', 'corte', 'momento'
listaV=[]
for i in range(len(max_vig_ei)):
    lista1=[]
    lista2=[]

```

```

        for j in range(len(max_vig_ei[i])):
            lista1=[[round(max_vig_si[i][j][1]/1000,2), round(max_vig_ei[i][j][1]/1000,2),
                    round(max(max_vig_ei[i][j][2],max_vig_si[i][j][2])/1000,2),
                    round(min(min_vig_ei[i][j][2],min_vig_si[i][j][2])/1000,2),round(max_vig_dli[i][j][1]/1000,2),
                    largosV[i][j],mat_vig_s[i][j][0],mat_vig_e[i][j][0]),
                    round(max_vig_sj[i][j][1]/1000,2), round(max_vig_ej[i][j][1]/1000,2),
                    round(max(max_vig_ej[i][j][2],max_vig_sj[i][j][2])/1000,2),
                    round(min(min_vig_ej[i][j][2],min_vig_sj[i][j][2])/1000,2),round(max_vig_dlj[i][j][1]/1000,2),
                    largosV[i][j],mat_vig_s[i][j][1],mat_vig_e[i][j][1])]
            lista2.append(lista1)
        listav.append(lista2)
    listaC=[]
    for i in range(len(max_col_ei)):
        lista1=[]
        lista2=[]
        for j in range(len(max_col_ei[i])):
            lista1=[[round(max(max_col_ei[i][j][0], max_col_si[i][j][0])/1000,2),
                    round(min(min_col_ei[i][j][0], min_col_si[i][j][0])/1000,2),
                    round(max_col_si[i][j][1]/1000,2),
                    round(max(max_col_ei[i][j][2],max_col_si[i][j][2]),
                    abs(min_col_ei[i][j][2]),abs(min_col_si[i][j][2])/1000,2),
                    largosC[i][j],mat_col_s[i][j][0],mat_col_e[i][j][0]),
                    round(max(max_col_ej[i][j][0], max_col_sj[i][j][0])/1000,2),
                    round(min(min_col_ej[i][j][0], min_col_sj[i][j][0])/1000,2),
                    round(max_col_sj[i][j][1]/1000,2),
                    round(max(max_col_ej[i][j][2],max_col_sj[i][j][2]),
                    abs(min_col_ej[i][j][2]),abs(min_col_sj[i][j][2])/1000,2),
                    largosC[i][j],mat_col_s[i][j][1],mat_col_e[i][j][1])]
            lista2.append(lista1)
        listaC.append(lista2)
    # lista_aux=[listaC[-1][j][:6] for j in range(len(listaC[0]))]
    # return [listav,listavmax, listaC, listaCmax]
    return [listav, listaC, exc]
    # return [listav, listaC, exc, lista_aux]

# print(filtroCV(combis, combi_e, combi_s, tab, largosv, largosC))

def v2vig(x1, lo, vuLsti, vueLsti, vuLstj, vueLstj, vupr, vc, state):
    vc = vc if state==1 else 0
    v2Calc = lambda v1, v2, x1, lo: round(v1 - x1 * (v1 - v2) / lo, 1)
    vupr2 = v2Calc(vupr,-vupr,x1,lo)/0.75-vc
    vu2 = max([v2Calc(vuLsti[i],vuLstj[i], x1, lo) for i in range(len(vuLsti))])/0.75-vc
    vue2 = max([v2Calc(vueLsti[i],vueLstj[i], x1, lo) for i in range(len(vueLsti))])/0.6
    return round(max(vupr2,vu2, vue2),1)

def b1(fc):
    if 550 >= fc >= 280:
        return round(0.85-0.05/70*(fc-280), 2)
    else:
        return 0.85 if fc < 280 else 0.65

def et(h,eu,dp,c): return round(eu*abs(h-dp-c)/c, 4)

def aCir(d): return round(0.007854*d**2, 3)

def phi(eu,et,ey):
    if ey <= et <= (eu+ey):
        return round(0.65+0.25/ey*(et-ey), 2)
    else:
        return 0.65 if et < ey else 0.9

def aLstC(dEsq,dLat,nHor,nVer):
    a = round(aCir(dEsq)*2+nHor*aCir(dLat), 3)
    return [a]+[round(aCir(dLat)*2,3) for i in range(nVer)]+[a]

def yLstC(dp,h,nVer):
    yLst = [dp]
    for i in range(1,nVer+1):
        yi = round((h-yLst[i-1]-dp)/(nVer+2-i)+yLst[i-1],0)
        yLst.append(int(yi))
    yLst.append(h-dp)
    return yLst

def pmC(aLst,b,b1,c,es,eu,ey,fc,fy,h,yLst):
    eiLst = [round(eu*(c-i)/c, 5) for i in yLst]
    fsLst = [fy*abs(i)/i if abs(i)>ey else es*i for i in eiLst]
    psLst = [fsLst[i] * aLst[i] for i in range(len(aLst))]
    pc = 0.85*b1*fc*b*c

```

```

Ps = sum(pSLst)
Mc = Pc/2*(h-0.85*c)
Ms = sum((pSLst[i]*(h/2-yLst[i]) for i in range(len(aLst))))
return [round((Pc+Ps)/1000, 2), round((Mc+Ms)/100000, 2)]

def cPn(aLst,b,b1,dp,es,eu,ey,fc,fy,h,pnB,yLst):
    c1 = 0
    c2 = max(h/b1, 3*(h-dp))
    PnMax = round((0.85*fc*(h*b-sum(aLst))+sum(aLst)*fy)/1000, 2)
    PhiPnMax = PnMax*0.8*0.65
    PnMin = round((-sum(aLst)*fy)/1000, 2)
    PhiPn = pnB+10
    i = 0
    if pnB > PnMin * 0.9:
        pnB = PhiPnMax if pnB >= PhiPnMax else pnB
        while abs(pnB-PhiPn) > 0.1 and i<15:
            c = round((c1+c2)/2,3)
            i += 1
            PMC = pmC(aLst,b,b1,c,es,eu,ey,fc,fy,h,yLst)
            eT = et(h,eu,dp,c)
            Phi = phi(eu,eT,ey)
            PhiPn = (PMC[0])*Phi
            PhiMn = (PMC[1])*Phi
            c2 = c if PhiPn > pnB else c2
            c1 = c if PhiPn < pnB else c1
    else:
        c = 0
        PhiPn = PnMin*0.9
        PhiMn = 0
        Phi = 0.9
    return [round(c, 2), abs(round(PhiMn, 1)), round(PhiPn, 1), Phi]

def cFind(aLst, b, b1, dp, es, eu, ey, fc, fy, h, mu, pu, yLst):
    mu = round(abs(mu),3)
    pu = round(pu,3)
    PhiPnMin = round((-sum(aLst)*fy)/1000*0.9,1)
    PhiPnMax = round((0.85*fc*(h*b-sum(aLst))+sum(aLst)*fy)*0.8*0.65/1000,1)
    if pu<PhiPnMin:
        pu = PhiPnMin
    if pu>PhiPnMax:
        pu = PhiPnMax
    elif abs(pu) <= 0.1:
        return cPn(aLst,b,b1,dp,es,eu,ey,fc,fy,h,0,yLst)
    e = min(mu/pu,999)
    i = 0
    c2 = 0
    ex = e+1
    c1 = h/b1 if e > 0 else cPn(aLst,b,b1,dp,es,eu,ey,fc,fy,h,0,yLst)[0]
    while abs(round(e,3)-ex) > 0.001 and i < 15:
        c = round((c1+c2)/2,2)
        i += 1
        PMC = pmC(aLst,b,b1,c,es,eu,ey,fc,fy,h,yLst)
        ex = round((abs(PMC[1]))/(PMC[0]),3)
        c1 = c if ex < e else c1
        c2 = c if ex > e else c2
    e = ex
    eT = round(eu*abs(h-dp-c)/c,4)
    Phi = phi(eu,eT,ey)
    asdf=pmC(aLst,b,b1,c,es,eu,ey,fc,fy,h,yLst)
    phipn = PMC[0]*Phi
    phimn = PMC[1]*Phi
    return [c,abs(round(phimn,1)),round(hipn,1), Phi, e, PhiPnMin, PhiPnMax]

def resumen(aLst, c, b, dp, h, eu, fy, fc, b1, es, ey, yLst):
    PMC = pmC(aLst, b, b1, c, es, eu, ey, fc, fy, h, yLst)
    eT = round(eu*abs(h-dp-c)/c, 4)
    Phi = phi(eu, eT, ey)
    PMCpr = pmC(aLst, b, b1, c, es, eu, ey, fc, fy*1.25, h, yLst)
    return [PMC[0]*Phi,PMC[0],PMCpr[0],PMC[1]*Phi,PMC[1],PMCpr[1]]

def FU(pu, mu, pn, mn):
    if abs(mu) < 0.1:
        return abs(pu/(pn+0.01))
    else:
        return max(abs(pu/(pn+0.01)), abs(mu/(mn+0.01)))

def yLstV(h, dp, db):
    # se busca el minimo de niveles barras laterales complementarias
    blat = min(int((h-2*dp-db/4)/25), int((h-2*dp-db/4)/20)+1)
    # se crea lista con dos primeros niveles (1/4*10=2.5 veces el diámetro mayor)
    Y = [dp, max(dp+db/4, 2*dp)]
    #se agrega cada nivel de barras complementarias
    for i in range(blat):
        Y.append(round(Y[-1]+(h-2*dp-db/4)/(blat+1), 0))
    # la función retorna la lista de posiciones de barras completa
    return Y + [h - dp] if Y[-1] < (h-dp) else Y

```

```

# función para el cálculo de área requerida asegurando  $et \geq 0.005$ 
def areav(mu, b, b1, h, fc, fy, dp):
    muu = round(mu/(0.9*0.85/100000*fc*b*(h-dp)**2), 3)
    muu = 0.5 if muu > 0.5 else muu
    ulim = round(0.375*b1*(1-0.1875*b1), 3)
    wp = 0 if muu < ulim else round((muu-ulim)/(1-dp/(h-dp)), 3)
    w = 0.375+wp if wp > 0 else round(1-(1-2*muu)**0.5, 3)
    return round(w*0.85*fc*b*(h-dp)/fy, 2)

def listadiam1(A, b, dp, h, dList, v):
    sup = [i for i in range(int(1+(b-2*dp)/15), 2+int((b-2*dp)/10))]
    listadiam = []
    for i in sup:
        n2 = int(i/2) if i>2 else 0
        n1 = i-n2
        for j in range(len(dList)):
            j = dList[j]
            if n2>0:
                for k in range(len(dList)):
                    k = dList[k]
                    if A<=n1*aCir(j)+n2*aCir(k) and j+v>=k>=j-v:
                        listadiam+=[[n1, j, n2, k, round(aCir(j)*n1+aCir(k)*n2, 2)]]
                    else:
                        continue
            else:
                if 1.2*A>=n1*aCir(j)>=A:
                    listadiam+=[[n1, j, n2, 0, round(n1*aCir(j), 2)]]
                else:
                    continue
    return listadiam

def listadiam(A, b, dp, h, dList, v):
    amin = 10*A
    A /= 2
    lista1 = listadiam1(A, b, dp, h, dList, v)
    lista2 = []
    minimos = []
    for i in range(len(lista1)):
        if lista1[i][4]<=1.2*A:
            lista2+=[lista1[i]]
        else:
            continue
    for i in range(len(lista2)):
        L1 = lista2[i]
        ar1= L1[4]
        ar2=round(2*A-ar1, 2)
        ar2 = ar2 if ar2>0 else 0
        lista3 = listadiam1(ar2, b, dp, h, dList, v)
        if lista3 == []:
            continue
        for j in range(len(lista2)):
            L2 = lista3[i]
            if 2*A>L1[4]+L2[4]:
                continue
            else:
                if L1[4]+L2[4]<amin:
                    amin = L1[4]+L2[4]
                    if L2[4]>L1[4]:
                        minimos = [L2, L1, round(amin, 2)]
                else:
                    minimos = [L1, L2, round(amin, 2)]
    return minimos

def critvc(vigas,columnas):
    newcol=[]
    for i in range(len(vigas)-1):
        new=[]
        for j in range(len(vigas[0])):
            if i == 0:
                mc1=columnas[i][j]
                mc2=columnas[i+1][j]
                mv1=vigas[i][j][0]
                mc=mc1+mc2
                dif=1.2*mv1-mc
                if dif > 0:
                    columnas[i][j]=dif*mc1/(mc1+mc2)+mc1
                    columnas[i+1][j]=dif*mc2/(mc1+mc2)+mc2
            elif i == len(vigas[0])-1:
                mc1 = columnas[i][j]
                mc2 = columnas[i + 1][j]
                mv2 = vigas[i][j][1]
                mc = mc1 + mc2
                dif = 1.2 * mv2 - mc
                if dif > 0:

```

```

        columnas[i][j] = dif * mc1 / (mc1 + mc2) + mc1
        columnas[i+1][j] = dif * mc2 / (mc1 + mc2) + mc2
    else:
        mc1 = columnas[i][j]
        mc2 = columnas[i+1][j]
        mv1 = vigas[i][j][0]
        mv2 = vigas[i][j][1]
        mc = mc1 + mc2
        dif = 1.2 * (mv1+mv2) - mc
        if dif > 0:
            columnas[i][j] = dif * mc1 / (mc1 + mc2) + mc1
            columnas[i+1][j] = dif * mc2 / (mc1 + mc2) + mc2
        newcol.append(new)
    newcol.append(columnas[-1])
    return newcol

def extMat(lista, indice):
    mat1=[]
    for i in lista:
        mat2=[]
        for j in i:
            mat2.append(j[indice])
        mat1.append(mat2)
    return mat1

def replMat(lista1, lista2, indice):
    for i in range(len(lista2)):
        for j in range(len(lista2[i])):
            lista1[i][j][indice]=lista2[i][j]
    return lista1

def matElemV(lista, bmaxv, hmaxv, ch, cs, b1, dp, es, ey, eu, fc, fy, dList, ai, deList, v):
    #se itera en la lista
    listav = []
    for i in lista:
        # se filtra la lista por piso
        tempv=[]
        for j in i:
            elem = optimusVig(j[2],j[3],es,eu,ey,b1,fc,fy,dp,dList,hmaxv,bmaxv,ai,j[5],ch,cs,v)
            tempv.append(elem)
        listav.append(tempv)
    return listav

def Lramas(xList):
    lar=len(xList)
    lista = [xList[0]]
    if lar%2 == 0:
        rango = xList[-1]-xList[0]
        for i in range(1, lar-1):
            if xList[i]-lista[-1]==30:
                lista.append(xList[i])
            elif xList[i]-lista[-1]>30:
                lista.append(xList[i-1])
        lista.append(xList[-1])
        minram = int((len(lista)+1)/2)*2
        maxram = len(xList)
        rlist = [i for i in range(minram, maxram+1, 2)]
        listas = []
        for i in rlist:
            sep = round(rango/(i-1), 1)
            complist = [xList[0]]
            for _ in range(i-1):
                complist.append(sep+complist[-1])
            lista2 = [xList[0]]
            for j in range(1, len(complist)-1):
                dif=999
                for k in xList:
                    if abs(k-complist[j])<dif:
                        dif = abs(k-complist[j])
                        bar = k
                lista2.append(bar)
            lista2.append(xList[-1])
            listas.append(lista2)
    else:
        mid = int(lar/2)
        midL = xList[0:mid+1]
        rango=midL[-1]-midL[0]
        for i in range(1, len(midL)-1):
            if midL[i]-lista[-1]==30:
                lista.append(midL[i])
            elif midL[i]-lista[-1]>30:
                lista.append(midL[i-1])
        lista.append(midL[-1])
        if lista[-2]-(xList[0]+xList[-1])/2<=15:
            lista.remove(lista[-1])
        lista2 = []

```

```

for j in reversed(lista):
    lista2.append(xList[-1]+xList[0]-j)
lista+=lista2
listas=[lista]
minram=2
maxram=len(xList)
rlist=[i for i in range(minram, maxram+1)]
for i in rlist:
    sep = round(rango/(i-1), 1)
    complist = [xList[0]]
    rango=xList[-1]-xList[0]
    for _ in range(i-1):
        complist.append(sep+complist[-1])
    lista2 = [xList[0]]
    for j in range(1, len(complist)-1):
        dif=999
        for k in xList:
            if abs(k-complist[j])<dif:
                dif = abs(k-complist[j])
                bar = k
        lista2.append(bar)
    lista2.append(xList[-1])
    if rlist[0]==i:
        continue
    else:
        listas.append(lista2)
listas2=[]
for i in listas:
    borrar = 0
    for j in range(1, len(i)):
        if i[j]-i[j-1]>30:
            borrar=1
    if borrar!=1:
        listas2.append(i)
return listas2

def estribosV(xList, ramas):
    Lestrib = []
    medio = xList[int(len(xList)/2)]
    for j in ramas:
        mid=int(len(j)/2)
        L1 = j[0:mid]
        if len(j)%2!=0:
            L2 = j[mid+1:]
            cond=1
        else:
            L2 = j[mid:]
            cond=0
        estribos=[[L1[i],L2[i]] for i in range(len(L1))]
        if cond==1:
            estribos+=[[medio]]
        Lestrib.append(estribos)
    return Lestrib

def ldc(fy,fc,db):
    return round(max(0.075*fy*0.1*db/(fc)**0.5, 0.0044*fy*0.1*db),1)

#traslapo vigas
def ldv(db, fc, fy):
    if db<19:
        return 0.1*db*fy/(21*(fc/10)**0.5)
    else:
        return 0.1*db*fy/(17*(fc/10)**0.5)

def ldhv(fy, db, fc):
    return fy*db/(170*(fc)**0.5)

def lGanchoC(db, fc, fy, h, dp):
    if db<19:
        ld = round(max(0.1*db*fy/(3.46*(fc)**0.5), 2.5*db+10),1)
        return [round(ldc(fy,fc,db)+0.6*3.1416/4*db+ld,1), ld]
    else:
        ld = round(max(0.1*db*fy/(4.4*(fc)**0.5), 2.5*db+10))
        return [round(ldc(fy,fc,db)+0.6*3.1416/4*db+ld,1), ld]

def lGanchoV(fy, db, fc):
    return round(ldhv(fy,db,fc)+1.05*db-5,1)

def rematC(db, ldv, h, dp):
    return max(2.5*db+10, ldv+dp-h)

def aminV(fc,b,fy):
    return max(0.2*(fc)**0.5*b/fy, 3.5*b/fy)

def countram(ramas):

```

```

nramas=[]
for i in ramas:
    nramas+=len(i)
return nramas

def Lest(h, b, dp, de):
    return round((2*(h+b-4*dp+0.2*de)*10+6.75*de*3.1416+2*max(75, 6*de))/10, 2)

def Ltrab(h, dp, de):
    return round((3.75*de*3.1416+2*max(75, 6*de)+(h+0.2*de-dp)*10)/10, 2)

def vc(fc, b, h, dp):
    return round(0.53*(fc)**0.5*b*(h-dp)/1000, 2)

def ashS(h, b, dp, fc, fy):
    return round(max(0.3*((b*h)/((h-dp)*(b-dp)))*(fc/fy), 0.09*(h-dp)*fc/fy), 3)

def loCol(h, b, H):
    return round(max(h, b, H/6, 45), 1)

def lEmp(fy, db):
    return round(max(0.00073*fy*db if fy<= 4200 else (0.0013*fy-2.4)*db, 30),0)

#wo es corte y no carga distribuida
def vprV(h, b, l, mpr1, mpr2, wo):
    return 100*(mpr1+mpr2)/l + wo/50

def vCAX(Nu, fc, b, h, dp):
    return round(0.53*(1+Nu*1000/(140*h*b))*(fc)**0.5*b*(h-dp)/1000, 1)

def vSLim(fc, b, h, dp):
    return round(2.2*(fc)**0.5*b*(h-dp)/1000,2)

def sRotV(h, dp, db):
    return round(max(min(15, 0.6*db, (h-dp)/4),8), 1)

def sRotC(h, b, db, hx):
    return round(min(max(min(15,0.6*db,(10+(35-hx)/3)),8),10),1)

def sMax(fc, b, h, dp, sm):
    return min(round((h-dp)/4 if vc(fc, b, h, dp)>0.33*(h-dp)*b*(fc/10)**0.5 else (h-dp)/2, 2), sm)

def sEmp(h, dp):
    return round(min(10, (h-dp)/4), 1)

def sCol(db):
    return min(0.6*db, 15)

def cubEstV(h, dp, de, Le):
    lista1 = []
    lista2 = []
    for i in Le:
        if len(i)%2 == 0:
            b = i[1]-i[0]
            lista1 += [Lest(h, b, dp, de)]
        else:
            lista2 += [Ltrab(h, dp, de)]
    return [round((sum(lista1)+sum(lista2))*acir(de) ,1), lista1, lista2]

def estribosC(xList):
    lista = []
    ramas = Lramas(xList)
    count = []
    for i in ramas:
        count.append(len(i))
        Lestrib = []
        temp = i
        while len(temp) > 0:
            if len(temp) >= 2:
                Lestrib.append([temp[0],temp[-1]])
                temp.remove(temp[0])
                temp.remove(temp[-1])
            elif len(temp) == 1:
                Lestrib.append([temp[0]])
                temp.remove(temp[0])
            else:
                break
        lista.append(Lestrib)
        Lestrib = []
    return lista, count

def xLst(sup, b, dp):
    mid = int(sup[0] / 2)
    if sup[2]%2==0:
        l1=[sup[1] for i in range(mid)]
        l2=[sup[3] for i in range(sup[2])]

```



```

else:
    l1=[sup[1] for i in range(mid)]
    l2=[sup[3] for i in range(sup[2])]
    lista=l1+l2+l1
    xList=yLstC(dp, b, len(lista)-2)
    return lista, xList

def minEstC(mpr1, mpr2, Nu, H, vu, vue, yList, deList, db, h, b, dp, fy, fc, cs, hvig):
    salida1, salida2, salida3 = 0, 0, 0
    H*=100
    vu = vu*1000
    vue = vue*1000
    mpr1*=100000
    mpr2*=100000
    Vc = VcAx(Nu, fc, b, h, dp)*1000
    vupr = round((mpr1+mpr2)/H,1)
    vupr1 = vupr if Nu*1000 < 0.05 * fc * (h * b) else vupr-vc
    vupr2 = vupr-vc
    vu1 = round(max((vu-vc)/0.75, vue/0.6, vupr1/0.75),1)
    vsLim = vsLim(fc, b, h, dp)*1000*1.1
    lo = locol(h, b, H)
    vu2 = round(max((vu-vc)/0.75, vue/0.6, (vupr1-vc)/0.75), 1)
    s = round(scol(db))
    estr = estribosC(yList)
    est = estr[0]
    nRam = estr[1]
    ramas = Lramas(yList)
    if len(ramas)>1:
        srotL = [int(sRotC(h, b, db, l)) for l in [min(k) for k in [[i[j]-i[j-1] for j in
range(1,len(i))] for i in ramas]]]
    else:
        ramas = ramas[0]
        aux1=[ramas[i]-ramas[i-1] for i in range(1,len(ramas))]
        srotL = [int(sRotC(h, b, db, max(aux1)))]

    sash = round(max(ashS(h, b, dp, fc, fy), aminV(fc,b,fy)), 3)

    s1L = [[i, j, k, l] for i in range(len(nRam)) for j in deList for l in deList if l <= j
        for k in range(8, min(int(sRotC(h, b, db, srotL[i])), int(round(100/((sash * 100 / (2 *
aCir(j) + (nRam[i] - 2) * aCir(1))-1), 1))+1))
        if vu1 <= round((2*aCir(j)+(nRam[i]-2)*aCir(1))*fy*(h-dp)/k, 1) <= vsLim]

    if s1L==[]:
        return 0
    minimo = 99999999
    for i, j, k, l in s1L:
        ramas1 = est[i]
        l1 = Lest(h, ramas1[0][1]-ramas1[0][0], dp, j)
        l2 = sum([Lest(h, ramas1[m][1]-ramas1[m][0], dp, l) if len(ramas1[m])==2 else Ltrab(h, dp, l)
            for m in range(1,len(ramas1))])
        s1 = int((lo-0.01)/k)+1
        costo = round(2*s1*(l1*aCir(j)+2*l2*aCir(1))*cs/1000000, 0)
        if costo<minimo:
            minimo=costo
            l2a = [Lest(h, ramas1[m][1] - ramas1[m][0], dp, l) if len(ramas1[m]) == 2 else Ltrab(h,
dp, l)
                for m in range(1, len(ramas1))]
            lram = ramas1
            lista1=[costo, nRam[i], j, k, l, s1, l1, l2a, l2, lram, lo]
            salida1=1
        l_rot = lista1[3]
        l_emp = lEmp(fy, db)
        s2L = [[i, j, k, l] for i in range(len(nRam)) for j in deList for l in deList if l <= j
            for k in range(10, min(int(s), int(round(100/((sash * 100 / (2 * aCir(j) + (nRam[i] - 2) *
aCir(1))-1), 1))+1))
            if vu2 <= round((2*aCir(j)+(nRam[i]-2)*aCir(1))*fy*(h-dp)/k, 1) <= vsLim]
        if s2L==[]:
            return 0
        minimo = 99999999
        for i, j, k, l in s2L:
            ramas1 = est[i]
            l1 = Lest(h, ramas1[0][1] - ramas1[0][0], dp, j)
            l2 = sum([Lest(h, ramas1[m][1] - ramas1[m][0], dp, l) if len(ramas1[m]) == 2 else Ltrab(h, dp,
l)
                for m in range(1, len(ramas1))])
            s2 = int((H-2*lo-l_emp-0.01)/k)
            dist2 = H-2*lo-l_emp
            costo = round(s2*(l1*aCir(j)+2*l2*aCir(1))*cs/1000000, 0)
            if costo<minimo:
                minimo=costo
                l2a = [Lest(h, ramas1[m][1] - ramas1[m][0], dp, l) if len(ramas1[m]) == 2 else Ltrab(h,
dp, l)
                    for m in range(1, len(ramas1))]
                lram = ramas1
                lista2=[costo, nRam[i], j, k, l, s2, l1, l2a, l2, lram, dist2]
                salida2=1

```

```

semp = int(sEmp(h, dp))
s3L = [[i, j, k, l] for i in range(len(nRam)) for j in deList for l in deList if l <= j
        for k in range(8, min(int(semp), int(round(100/((sash * 100 / (2 * acir(j) + (nRam[i] - 2)
* acir(l))))-1), 1)+1))
        if vu2 <= round((2*acir(j)+(nRam[i]-2)*acir(l))*fy*(h-dp)/k, 1) <= vslim]
if s3L==[]:
    return 0
minimo = 99999999
for i, j, k, l in s3L:
    ramas1 = est[i]
    l1 = Lest(h, ramas1[0][1] - ramas1[0][0], dp, j)
    l2 = sum([Lest(h, ramas1[m][1] - ramas1[m][0], dp, l) if len(ramas1[m]) == 2 else Ltrab(h, dp,
1)
                for m in range(1, len(ramas1))])
    s3 = int((l_emp-0.01)/k)+1
    costo = round(s3*(l1*acir(j)+2*l2*acir(l))*cs/1000000, 0)
    if costo < minimo:
        minimo = costo
        l2a = [Lest(h, ramas1[m][1] - ramas1[m][0], dp, l) if len(ramas1[m]) == 2 else Ltrab(h,
dp, l)
                for m in range(1, len(ramas1))]
    lram = ramas1
    lista3 = [costo, nRam[i], j, k, l, s3, l1, l2a, l2, lram, l_emp]
    salida3=1
    costo_total = lista1[0]+lista2[0]+lista3[0]
    # lista1 --> [costo, n° ramas, de_externo, espaciamiento, de_interno, n° estribos, largo1, largos2,
largo_tot2, d_ramas, dist]
    # lista2 --> [costo, n° ramas, de_externo, espaciamiento, de_interno, n° estribos, largo1, largos2,
largo_tot2, d_ramas, dist]
    # lista3 --> [costo, n° ramas, de_externo, espaciamiento, de_interno, n° estribos, largo1, largos2,
largo_tot2, d_ramas, dist]
    salida=salida1+salida2+salida3
    if salida == 3:
        return [lista1,lista2,lista3,costo_total,vu1,vu2,hvig]
    else:
        return 0

def optimusCol(b1, dp, es, eu, ey, fc, fy, muC, muCmin, puCmin, puCmax, dList, hmax,
               hmin, ch, cs, H, vu, vue, deList, iguales, hvig):
    salida=0
    minor = 99999999
    hmin = hmin if hmin >= 30 else 30
    hmax = hmax if hmax >= 30 else 30
    hList = [i for i in range(hmin, hmax+5,5)]
    lista = ([b, h] for b in hList for h in hList if b == h)
    for b, h in lista:
        nH = [i for i in range(int((b-2*dp)/15)-1, int(round((b-2*dp)/10, 0)), 1)]
        nv = nH
        listaND = ([j, k] for j in nH for k in nv if 10 <= (b-2*dp)/(j+1) <= 15 and
10 <= (h-2*dp)/(k+1) <= 15 and j == k)
        for j, k in listaND:
            if iguales == 0:
                listaDm = ([l, m] for l in dList for m in dList if m <= l >=16)
            else:
                listaDm = ([l, m] for l in dList for m in dList if m == l >= 16)
            for l, m in listaDm:
                ylist = yLstC(dp, h, k)
                alist = aLstC(l, m, j, k)
                cF = cFind(alist, b, b1, dp, es, eu, ey, fc, fy, h, muC, puCmax, ylist)
                cF2 = cFind(alist, b, b1, dp, es, eu, ey, fc, fy, h, muCmin, puCmin, ylist)
                fu = round(FU(puCmax, muC, cF[2], cF[1])*100,1)
                fu2 = round(FU(puCmin, muCmin, cF2[2], cF2[1])*100,1)
                as = acir(l)*4+acir(m)*(2*j+2*k)
                cuan = round(as/(b*h), 5)
                mpr1 = max(puC(alist, b, b1, cF[0], es, eu, ey, fc, fy*1.25, h, ylist)[1],
                           puC(alist, b, b1, cF2[0], es, eu, ey, fc, fy*1.25, h, ylist)[1])
                mpr2 = mpr1
                #agregar a entrada H, vu, vue, deList
                if fu < 95 and fu2 < 95 and 0.01 <= cuan <= 0.06:
                    cortel = minEstC(mpr1, mpr2, muC, H, vu, vue, ylist,
                                     deList, min(l, m), h, b, dp, fy, fc, cs, hvig)
                    if cortel != 0:
                        costo1 = round((as*cs+(b*h-as)*ch)/10000, 0)*(cortel[2][10]+H*100)/100
                        costo2 = cortel[3]
                        costo = costo1+costo2
                        if costo < minor:
                            # corte = minEstC(mpr1, mpr2, muC, H, vu, vue, ylist, deList, min(l, m),
h, b, dp, fy, fc, cs)
                            minor, e = costo, round(cF[1] / (cF[2] + 0.001), 3)
                            optimo = [minor, h, b, j, k, l, m, fu, fu2, cuan, cF[0], cF2[0], e, alist,
                                     cF[2], muC, puCmax, puCmin, H, iguales, round(muCmin/puCmin,3),
cF2[1], cF2[2], costo1, costo2, dp]
                            salida=1
                            corte=cortel
            else:

```

```

        continue

    if salida==1:
        return [optimo, corte]
    else:
        return 0

def minEstV(mpr1, mpr2, vuLsti,vueLsti,vuLstj,vueLstj, xList, deList, db, h, b, lo, dp, fy, fc, cs,
wo, yLst,hcol, emp1,emp2):
    lo*=100
    Vc = vc(fc, b, h, dp)*1000
    vupr = round(vprv(h, b, lo, mpr1, mpr2,wo),3)*1000
    smax = sMax(fc, b, h, dp, 20)
    srot = int(sRotV(h, dp, db))
    sL1 = [i for i in range(8, int(srot)+1)]
    sL2 = [i for i in range(8, int(smax)+1)]
    vsL = vsLim(fc, b, h, dp)*1000*1.1
    ramas = Lramas(xList)
    est = estribosV(xList, ramas)
    nRam = countram(ramas)
    x1 = 2*h
    x2 = lo/2-2*h
    amin=aminV(fc,b,fy)
    Lout=[]
    for n in range(x1, x1 + 5, 5):
        xal = n
        xa2 = (x1 + x2) - xal
        vsB1 = V2vig(0,lo,vuLsti,vueLsti,vuLstj,vueLstj,vupr,Vc,0)
        vsB2 = V2vig(xal,lo,vuLsti,vueLsti,vuLstj,vueLstj,vupr,Vc,1)
        lista=[i,j,k,l,m] for i in nRam for j in sL1 for k in deList for l in nRam
        for m in sL2 if vsB1/(fy*(h-dp))<=i*aCir(k)/j<=vsL/(fy*(h-dp))
        and vsB2/(fy*(h-dp))<=l*(aCir(k))/m<=vsL/(fy*(h-dp)) and i*aCir(k)>amin]
        minim = 999999999
        if lista!=[]:
            for i in lista:
                nr1, s1, de, nr2, s2 = i
                Lest1 = est[nRam.index(nr1)]
                Lest2 = est[nRam.index(nr2)]
                LestH = Ltrab(b, dp, de)
                ns1=int((xa1*2)/s1)
                if s2>10:
                    ns2=int((xa2-(emp1+emp2)/2-0.01)*2/s2)+1
                    ns3=int(((emp1+emp2)-0.01)/10)
                    ns4 = ns2 if ns2>0 else 0
                    ns2=ns4+ns3
                else:
                    ns2 = int((xa2-0.01)*2 / s2) + 1
                nsH=ns1+ns2
                numH=len(yLst)-2
                cub1=cubEstV(h, dp, de, Lest1)
                cub2=cubEstV(h, dp, de, Lest2)
                mini = (cub1[0]*ns1+cub2[0]*ns2+LestH*nsH*numH)*cs/1000000
                x1 = xal-5 if xal > 2*h else 2*h
                x2 = 2*((x1+x2)-x1)
                x3 = emp1+emp2
                x2 = round(x2-x3,1) if x2-x3>0 else 0
                # if s2 > 10:
                if mini < minim:
                    minim = round(mini, 2)
                    #[costo, dist rot, n° ramas, espaciamento, n° estribos, dist de rotula al centro,
n° ramas, espaciamento, n° estribos, de]
                    Lout = [minim, x1, nr1, s1, ns1, x2, nr2, s2, ns2, de, vsB1, vsB2, cub1, cub2,
nsH, numH,
                        LestH,x3,emp1,emp2]

    return Lout

def optimusVig(mpp,mnn,es,eu,ey,b1,fc,fy,dp,dList,dimV,ai,lo,CH,CS,v,allVu,deList,wo,nbahias,hcol):
    lo=hcol/100
    di = round((ai*200/3.1416)**0.5,1)
    mnn=abs(mnn)
    salida=0
    minim = 999999999
    hmax = dimV[0] if dimV[0]>=30 else 30
    bmax = dimV[1] if dimV[1]>=25 else 25
    hmin = dimV[2] if dimV[2]>=30 else 30
    bmin = dimV[3] if dimV[3]>=25 else 25
    hList = [i for i in range(hmin, hmax+5,5)]
    bList = [i for i in range(bmin, bmax+5,5)]
    lista = ([i, j] for i in hList if i >= 100*lo/16 for j in bList if i >= j and j >= 0.4*i)
    for h, b in lista:
        A1 = areaV(mpp, b, b1, h, fc, fy, dp)
        # print(A1)
        A2 = areaV(mnn, b, b1, h, fc, fy, dp)
        # print(A2)
        L1 = listadiam(A1, b, dp, h, dList, v)
        if L1==[]:

```

```

        continue
L2 = listadiam1(A2, b, dp, h, dList, v)
mi1 = 10*A2
lis=[]
for i in range(len(L2)):
    L=L2[i]
    if L[4]<mi1:
        mi1=L[4]
        lis = L
if lis==[]:
    continue
db = max(L1[0][1], L1[0][3])
db2 = max(L1[1][1], L1[1][3])
db3 = di
db4 = max(L2[0][1], L2[0][3])
ganchol = round(lGanchov(fy, db, fc),1)
gancho2 = round(lGanchov(fy, db2, fc),1)
gancho3 = round(lGanchov(fy, db3, fc),1)
gancho4 = round(lGanchov(fy, db4, fc),1)
traslp1=round(ldv(db, fc, fy)*1.3,1)
traslp2 = round(ldv(db2, fc, fy) * 1.3, 1)
traslp3=round(ldv(db3, fc, fy),1)
traslp4=round(ldv(db4, fc, fy),1)
ldh1 = round(ldhv(fy,db,fc),1)
ldh2 = round(ldhv(fy, db2, fc),1)
ldh3 = round(ldhv(fy, db3, fc),1)
ldh4 = round(ldhv(fy, db4, fc),1)
suple1 = max(0.25*lo*100,ldv(db2, fc, fy))
suple2 = max(0.3*lo*100,ldv(db2, fc, fy))
volSup = (2*(suple1+gancho2)+(nbahias-1)*(suple2+hcol))*L1[1][4]
volBar = L1[0][4]*(nbahias*lo*100+(nbahias-1)*hcol+2*ganchol+1Emp(fy,db))\
+ai*(nbahias*lo*100+(nbahias-1)*hcol+2*gancho3+1Emp(fy,db3))\
+lis[4]*(nbahias*lo*100+(nbahias-1)*hcol+2*gancho4+1Emp(fy,db4))
lDetail = [[db, 1.2*db, ganchol, traslp1, ldh1],
            [db2, 1.2*db2, gancho2, traslp2, ldh2],
            [db3, 1.2*db3, gancho3, traslp3, ldh3],
            [db4, 1.2*db4, gancho4, traslp4, ldh4],
            [suple1, suple2, volSup, volBar, hcol]]
ylst = list(yLstV(h, dp, db))
ylstrev = [(h-i) for i in reversed(ylst)]
aSLst = [L1[0][4], L1[1][4]]+[ai for i in range(len(ylst)-3)]+[lis[4]]
alstrev = [lis[4]]+[ai for i in range(len(ylst)-3)]+[L1[1][4], L1[0][4]]
cuanT = round(sum(aSLst)/(h*b-sum(aSLst)), 4)
cumin = round(max(0.8/fy*(fc*0.5), 14/fy), 4)
cuan1 = round((aSLst[0]+aSLst[1])/((b*(h-dp))), 4)
cuan2 = round(aSLst[-1]/((b*(h-dp))), 4)
cpn = cPn(aSLst, b, b1, dp, es, eu, ey, fc, fy, h, 0, ylst)
cpnrev = cPn(alstrev, b, b1, dp, es, eu, ey, fc, fy, h, 0, ylstrev)
c = cpn[0]
cond = False
eT = round(eu*abs(h-dp-c)/c, 4)
mpr1 = pmC(aSLst, b, b1, cpn[0], es, eu, ey, fc, fy * 1.25, h, ylst)[1]
mpr2 = pmC(alstrev, b, b1, cpnrev[0], es, eu, ey, fc, fy * 1.25, h, ylstrev)[1]
db = min([L1[0][1] if L1[0][1]>0 else 99
          ,L1[0][3] if L1[0][3]>0 else 99
          ,lis[1] if lis[1]>0 else 99
          ,lis[3] if lis[3]>0 else 99])
sup=L1[0]
xlistV = xLst(sup, 30, 5)[1]
FU = round(max(mnn / cpn[1], mpp / cpnrev[1]) * 100, 1)
if 0.025 >= cuan1 >= cumin and 0.025 >= cuan2 >= cumin\
and cpn[1] >= mnn and cpnrev[1] >= mpp and 85<=FU<=95:
    cond = True
    cortel = minEstV(mpr1, mpr2, allVu[0], allVu[1], allVu[2], allVu[3], xlistV, deList, db,
                    fy, fc, cs, wo, ylst,hcol,traslp1,traslp4)
    costo = round((volSup + volBar) * cs / 1000000 + (h * b) * nbahias * cH / 10000 +
cortel[0]*nbahias, 0)
    if costo < minim and cond != False:
        minim = costo
        FU = round(max(mnn/cpn[1], mpp/cpnrev[1]) * 100, 1)
        corte=cortel
        listaT = [minim, h, b, aSLst, ylst, cuan1, cuan2, ylstrev, alstrev,c ,
round(abs(mnn),2), round(abs(mpp),2), L1, lis, cpn[1], cpnrev[1], max(cpn[1],cpnrev[1]), lo,
FU, lDetail, xlistV]
        salida = 1
    if salida == 1:
        return [listaT, corte]
    else:
        return 0

def XYplotCurv(alst, b, h, dp, eu, fy, fc, b1, es, ey, ylst, ce, mu, pu, mn, pn, titulo):
    PnMax = round((0.85*fc*(h*b-sum(alst))+sum(alst)*fy)/1000, 2)
    PnMaxPr = round(PnMax+sum(alst)*fy*0.25/1000, 2)
    PnMin = sum(alst)*-fy/1000

```

```

phiPnMin = 0.9*sum(alst)*-fy/1000
PnMinPr = 1.25*sum(alst)*-fy/1000
C = [0]+[i/50*h for i in range(2, 51)]
X1 = [0]
X2 = [0]
X3 = [0]
Y1 = [phiPnMin]
Y2 = [PnMin]
Y3 = [PnMinPr]
for c in C[1:]:
    res = resumen(alst, c, b, dp, h, eu, fy, fc, b1, es, ey, ylst)
    X1.append(res[3])
    Y1.append(res[0])
    X2.append(res[4])
    Y2.append(res[1])
    X3.append(res[5])
    Y3.append(res[2])
X1.append(0)
X2.append(0)
X3.append(0)
Y1.append(Y1[-1])
Y2.append(PnMax)
Y3.append(PnMaxPr)
fig = plt.figure(figsize=[4,6], dpi=200)
plt.plot(X1, Y1, label='ØMn - ØPn', color='steelblue')
plt.plot(X2, Y2, label='Mn - Pn', color='crimson')
plt.plot(X3, Y3, label='Mpr - Ppr', color='forestgreen')
plt.plot([mu], [pu], marker='x', markersize=10, color='red', label='Mu - Pu', lw='1')
res1 = resumen(alst, ce, b, dp, h, eu, fy, fc, b1, es, ey, ylst)
plt.plot([0, mu], [0, pu], ls='--', color='black')
# plt.plot([mu, mn], [pu, pn], ls='--', color='gray')
plt.xlabel('Mn[tonf-m]')
plt.xlim([0, max(X3)+0.1])
plt.ylabel('Pn[tonf]')
plt.title('titulo')
plt.legend()
plt.grid()
# plt.show()
fig.savefig(titulo)
return 0

hminv = 25
bminv = 25
hmaxv = 70
bmaxv = 50

ncol = 16
cvig = 12

listadim=p.tabla['perfiles']

# print(listadim)
npisos=4
nbahias=3
# dimCol = [[[listadim[ncol][(i)+j*(nbahias+1)]]],listadim[ncol][(i)+j*(nbahias+1)]] for i in
range(nbahias+1)] for j in range(npisos)]

dimv = [[[int(round((listadim[ncol][(i)+j*(nbahias+1)]])/5,1))*5+15,
int(round((listadim[ncol][(i)+j*(nbahias+1)]])/5,1))*5+15,
int(round((listadim[ncol][(i)+j*(nbahias+1)]])/5,1))*5-5,
int(round((listadim[ncol][(i)+j*(nbahias+1)]])/5,1))*5-5]
for i in range(nbahias+1)] for j in range(npisos)]
dimC = [[[int(round((listadim[ncol][(i)+j*(nbahias+1)]])/5,1))*5+15,
int(round((listadim[ncol][(i)+j*(nbahias+1)]])/5,1))*5+15,
int(round((listadim[ncol][(i)+j*(nbahias+1)]])/5,1))*5,
int(round((listadim[ncol][(i)+j*(nbahias+1)]])/5,1))*5]
for i in range(nbahias+1)] for j in range(npisos)]

# print(listadim[ncol:])
# print(dimV,"\\n\\n",dimC)

largosc=[[3,3,3,3],
[3,3,3,3],
[3,3,3,3],
[3,3,3,3]]

largosv=[[7,7,7],
[7,7,7],
[7,7,7],
[7,7,7]]

# dimV = [[[65,40,25,25],[65,40,25,25],[65,40,25,25]],
# [[65,40,25,25],[65,40,25,25],[65,40,25,25]],
# [[65,40,25,25],[65,40,25,25],[65,40,25,25]],
# [[65,40,25,25],[65,40,25,25],[65,40,25,25]]]

```

```

def dimv(hmax,bmax,hmin,bmin,npisos,nbahias):
    return [[hmax,bmax,hmin,bmin]for i in range(nbahias)]for j in range(npisos)]

# listadimv = listadim
# dimv = dimv(65,40,25,25,3,4)

hcolMax, hcolMin = 90, 30
# dimv = dimv(65,40,25,25,4,3)

nbahias=len(largosv[0])

def matElemV(lista, ch, cs, bl, dp, es, ey, eu, fc, fy, dList, ai, deList, v, nbahias, dimC):
    #se itera en la lista
    listav = []
    for i in range(len(lista)):
        # se filtra la lista por piso
        tempv=[]
        for j in range(len(lista[0])):
            ultimo = 1 if i == len(lista)-1 else 0
            elem = optimusvig(lista[i][j][0],lista[i][j][1],es,eu,ey,b1,fc,fy,dp,dList,
                             lista[i][j][5],ai,lista[i][j][3],ch,cs,v,lista[i][j][4],
                             deList,lista[i][j][2],nbahias,dimC[i][1][0])

            cont=0
            while elem == 0 and cont<10:
                cont+=1
                lista[i][j][0]=lista[i][j][0]*1.1
                lista[i][j][1]=lista[i][j][1]*1.1
                elem = optimusvig(lista[i][j][0], lista[i][j][1], es, eu, ey, b1, fc, fy, dp, dList,
                                lista[i][j][5], ai, lista[i][j][3], ch, cs, v, lista[i][j][4],
                                lista[i][j][2],nbahias,dimC[i][1][0])

            tempv.append(elem)
            listav.append(tempv)
    return listav

ch, cs, b1, dp, es, ey, eu, fc, fy = 75000,7850000,0.85,5,2100000,0.002,0.003,250,4200
dList, deList = [16,18,22,25,28,32,36],[10,12,16]

def detvig(detvig, nbahias, hcol):
    # agregar lista de barras horizontales
    di = 12
    ai = 2.26
    contv = 0
    npisos=len(detvig)
    print("Reporte de diseño y cubicación de la estructura")
    print("\ncantidad pisos",npisos)
    print("cantidad vigas tipo por piso",len(detvig[0]), "\n\n")
    acum=0
    for i in detvig:
        for j in i:
            hcol=hcol[contv][0]

            """ Identificador """

            contv+=1
            print("Viga n° ",contv)
            print("Viga tipo del piso", contv,"\n\n")

            """Dimensiones"""

            print("Dimensiones\n")
            print("Largo : ", j[0][17], "[m]")
            print("Alto : ",j[0][1], "[cm]")
            print("Ancho : ",j[0][2], "[cm]\n")

            """Refuerzo longitudinal"""

            print("Refuerzo longitudinal\n")
            print("Armadura superior principal")
            numB2=" barras" if j[0][12][0][2]>1 else " barra"
            barr2 = "" if j[0][12][0][2]==0 else " "+str(j[0][12][0][2])+str(numB2)+" Ø "+\
                str(j[0][12][0][3])+"[mm] en la posición y = "+\
                str(j[0][4][0])+"[cm], área = "+str(j[0][3][0])+
                "[cm2]"

            print(j[0][12][0][0],"barras Ø",j[0][12][0][1],"[cm]",barr2)

            print("\nArmadura suplementaria")
            numB3 = " barras" if j[0][12][0][2] > 1 else " barra"
            barr3 = "" if j[0][12][1][2] == 0 else " "+ str(j[0][12][1][2])+str(numB3)+"\
                " Ø "+str(j[0][12][1][3])+"[mm] en la posición y =
                "+\

```

```

[cm2]"
str(j[0][4][1])+ " [cm], área = "+str(j[0][3][1])+
print(j[0][12][1][0], "barras Ø", j[0][12][1][1], "[cm]", barr3)
if len(j[0][3])>3:
    print("\nArmadura lateral")
    for i in range(len(j[0][3])-3, len(j[0][3])-1):
        print("2 barras Ø", di, "[mm] en la posición y = ", j[0][4][i], "cm, área =
", ai, "[cm2]")
    print("\nArmadura inferior principal")
    numB4 = " barras" if j[0][13][2] > 1 else " barra"
    barr4 = "" if j[0][13][2] == 0 else ", " + str(j[0][13][2])+str(numB3)+" Ø
"+str(j[0][13][3])+\"
        "[mm] en la posición y = "+str(j[0][4][-1])+ " [cm],
área = "+\
        str(j[0][3][-1])+ " [cm2]"
    print(j[0][13][0], "barras Ø", j[0][13][1], "[cm]", barr4, "\n")

    """Cuantías"""
    print("\nCuantías")
    print("Superior = ", j[0][5])
    print("Inferior = ", j[0][6], "\n")

    det=j[0][19]

    print("Cubicación de acero en barras longitudinales.\n")

    print("Largos de suples : ")

    print("0.25lo : ", round(det[4][0],1), "[cm]")
    print("0.3lo : ", round(det[4][1],1), "[cm]")
    print("kg de barras de suples : ", round(det[4][2]*0.007850,1), "[kg]")
    print("kg de otras barras : ", round(det[4][3] * 0.007850,1), "[kg]\n")
    vHorm=j[0][1]*j[0][2]*nbahias*(j[0][17])/10000
    print("Volumen de hormigón : ", vHorm, "[m3]\n")
    cAc=round(det[4][2]*0.007850+det[4][3] * 0.007850,1)*1000
    print("Costo de acero: $", cAc)
    cHorm=75000 * j[0][1] * j[0][2] * nbahias * (j[0][17]) / 10000
    print("Costo de hormigón : $", cHorm, "\n")

    # rem = 0 if contv < npisos else round(j[0][25]-j[0][21],1)

    print("Barras superiores : \n")

    print("Longitud de traslapo de armadura superior : ", det[0][3], "[cm]")
    print("Desarrollo de gancho : ", det[0][4], "[cm]")
    print("Gancho : ", det[0][2], "[cm]")
    print("Diámetro : ", det[0][0], "[mm]")
    print("12db : ", det[0][1], "[cm]\n")

    print("Barras suplementarias : \n")

    print("Desarrollo de gancho : ", det[1][4], "[cm]")
    print("Gancho : ", det[1][2], "[cm]")
    print("Diámetro : ", det[1][0], "[mm]")
    print("12db : ", det[1][1], "[cm]\n")

    print("Barras laterales : \n")

    print("Longitud de traslapo de armadura lateral : ", det[2][3], "[cm]")
    print("Desarrollo de gancho : ", det[2][4], "[cm]")
    print("Gancho : ", det[2][2], "[cm]")
    print("Diámetro : ", det[2][0], "[mm]")
    print("12db : ", det[2][1], "[cm]\n")

    print("Barras inferiores : \n")

    print("Longitud de traslapo de armadura inferior : ", det[3][3], "[cm]")
    print("Desarrollo de gancho : ", det[3][4], "[cm]")
    print("Gancho : ", det[3][2], "[cm]")
    print("Diámetro : ", det[3][0], "[mm]")
    print("12db : ", det[3][1], "[cm]\n")

    """Refuerzo transversal"""

    print("Refuerzo transversal")
    print("\nZonas de rótula plástica, de 0 a", j[1][1], "[cm] y ", j[0][17]*100-
j[1][1], "a", j[0][17]*100, "[cm]:")
    print("Diámetro : ", j[1][9], "[cm]")
    print("N° ramas : ", j[1][2])
    cont=0
    print("Estribos =", len(j[1][12][1]))
    for i in j[1][12][1]:
        cont+=1

```

```

        print("Largo de estribo n",cont,"=",i,"[cm]")
    if j[1][12][2]!=[]:
        print("Traba central: si")
        print("Largo de traba =",j[1][12][2][0],"[cm]")
    else:
        print("Traba central: no")
        print("Espaciamiento : ",j[1][3],"[cm]")
        print("N° estribos : ",int(round(j[1][4]/2,0)), " en cada extremo")
        # volt=
        print("Volumen de acero en zona de rótulas plásticas : ",
round(j[1][12][0]*j[1][4],1),"[cm3]")
        p1=round(j[1][12][0]*j[1][4]*0.00785,1)
        print("Peso del acero", p1,"[kg]\n")

        print("\nzonas central, de ",j[1][1],"a",j[0][17]*100-j[1][1],"[cm]:")
        print("Diámetro : ", j[1][9],"[cm]")
        print("N° ramas : ", j[1][6])
        print("longitud de empalme barras inferiores : ",j[1][18],"[cm]")
        print("longitud de empalme barras superiores : ",j[1][19],"[cm]")
        print("espaciamiento normal : ",j[1][7],"[cm]")
        print("espaciamiento zona de empalme : 10[cm]")

    cont=0
    for i in j[1][13][1]:
        cont+=1
        print("Largo de estribo n",cont,"=",i,"[cm]")
    if j[1][13][2]!=[]:
        print("Traba central: si")
        print("Largo de traba =",j[1][13][2][0],"[cm]")
    else:
        print("Traba central: no")
        print("Espaciamiento : ", j[1][7],"[cm]")
        print("N° estribos : ", j[1][8])
        print("Volumen de acero en zona central : ", j[1][13][0] * j[1][8], "[cm3]")
        p2=round(j[1][12][0] * j[1][8] * 0.00785,1)
        print("Peso del acero", p2,"[kg]\n")

        print("Trabas Horizontales")
        print("N° Trabas por estribo = ",j[1][15])
        print("N° de estribos donde va traba = ",j[1][14])
        print("Largo trabas = ",j[1][16],"[cm]")
        print("Volumen de acero en trabas horizontales : ",
round(j[1][15]*j[1][16]*ai*0.5,1),"[cm3]")
        p3=round(j[1][15]*j[1][14]*j[1][16]*ai*0.5*0.00785,1)
        print("Peso del acero : ", p3,"[kg]\n")

        print("Acero total : ",round(p1+p2+p3,1),"[kg]")
        print("Hormigón total : ",round(vHorm,2),"[m3]")
        print("Costo hormigón : $", cHorm)
        print("Costo acero : $", cAc+(p1+p2+p3)*1000)
        print("Costo total de vigas por piso : $",cAc+(p1+p2+p3)*1000+cHorm)
        acum+=cAc+(p1+p2+p3)*1000+cHorm

        # lista = [minim0, h1, b2, asLst3, ylst4, cuan1 5, cuan2 6, ylstrev7, alstrev8, c9,
round(abs(mnn), 2)10,
        # round(abs(mpp), 2)11, L1 12, lis 13, cpn[1]14, cpnrev[1]15, max(cpn[1],
cpnrev[1])16, lo17, FU18,
        # db2 19, gancho20, db12 21, nbarG 22, traslp1 23, traslp2 24, remate25, db122
26, remate2 27, gancho2 28, xlistV 29]
        # lout = [minim0, x11, nr12, s13, ns41, x52, nr26, s27, ns28, de9, vsB110, vsB211, cub112,
cub213,
        # nsH14, numH15,
        # LestH16, x317, emp118, emp219]

        """"Resultados""""

        print("Resultados del análisis\n")
        print("Flexión")
        print("ØMn+ = ", j[0][15], "[tf-m]")
        print("ØMn- = ", -j[0][14], "[tf-m]")
        print("F.U. mayor = ", j[0][18], "%\n")

        print("Corte")
        phiVn1 = round(acir(j[1][9])*j[1][2]*fy*(j[0][1]-dp)/j[1][3],1)
        print("Øvn1 = ",round(phiVn1/1000,1), "[tf]")
        fuV1 = round(100*j[1][10]/(phiVn1),1)
        print("F.U.1 = ",fuV1, "%")
        phiVn2 = round(acir(j[1][9])*j[1][6]*fy*(j[0][1]-dp)/j[1][7], 1)
        print("Øvn2 = ",round(phiVn2/1000,1), "[tf]")
        fuV2 = round(100*j[1][11]/phiVn2,1)
        print("F.U.2 = ",fuV2,"%\n")
        print("\n")
    return acum

def detcol(detcol):

```



```

print("\nNota: todas las columnas son simétricas, por lo tanto, su ancho y alto es igual.")
print("Por otro lado, las trabas y/o estribos interiores perpendiculares al eje x se replican al
eje y")
cont = 0
npisos = len(detcol)
ncol = len(detcol[0])
acum=0
for i in detcol:
    for j in i:
        """ Identificador """

        cont+=1
        piso=npisos if cont%npisos==0 else cont%npisos
        tipo = 2 if cont>npisos else 1

        print("\n\nColumna n° ",cont)
        print("Piso N°",piso)
        print("Tipo",tipo,"\n\n")

        """Dimensiones"""

        print("Dimensiones\n")
        print("Largo : ", j[0][20], "[m]")
        print("Alto : ", j[0][1], "[cm]")
        print("Ancho : ", j[0][2], "[cm]\n")

        """Refuerzo longitudinal"""

        print("Refuerzo longitudinal\n")
        list = []
        if j[0][21]!=1:
            print("Armadura superior")
            if j[0][3]>0:
                print("2 barras Ø",j[0][5],"[mm] y",j[0][3],"barras Ø",j[0][6],"[mm] en la
posición y =",j[0][14][0], "[cm], área =",j[0][13][0],"[cm2]")
            else:
                print("2 barras Ø",j[0][5],"[mm] en la posición y =",j[0][14][0],"[cm], área
=",j[0][13][0],"[cm2]")
            else:
                print(2+j[0][3],"barras Ø",j[0][5],"[mm] en la posición y =",j[0][14][0],"[cm], área
=",j[0][13][0],"[cm2]")
                if j[0][4]>0:
                    for i in range(j[0][4]):
                        print("2 barras Ø",j[0][6],"[mm] en la posición y =",j[0][14][i+1],"[cm], área
=",j[0][13][i+1],"[cm2]")
                if j[0][21]!=1:
                    print("Armadura superior")
                    if j[0][3]>0:
                        print("2 barras Ø",j[0][5],"[mm] y",j[0][3],"barras Ø",j[0][6],
"[mm] en la posición y =",j[0][14][-1],"[cm], área =",j[0][13][-1],"[cm2]")
                    else:
                        print("2 barras Ø",j[0][5],"[mm] en la posición y =",j[0][14][-1],"[cm], área
=",j[0][13][-1],"[cm2]")
                    else:
                        print(2+j[0][3],"barras Ø",j[0][5],"[mm] en la posición y =",j[0][14][-1],"[cm], área
=",j[0][13][-1],"[cm2]")

        """Cuantía"""

        print("\nCuantía = ",j[0][9],"[cm]\n")

        """Uniones y remates"""
        print("Uniones y remates\n")

        if piso!=npisos and piso>1:
            if cont>npisos:
                print("Columna para zonas centrales\n")
                print("Para unión superior\n")
                ldc = ldc(fy, fc, j[0][5])
                print("Longitud de empalme unión viga-columna = ", ldc, "[cm]\n")
                print("Para unión inferior\n")
                print("Longitud de gancho-remate = ", lG, "[cm]")
                ldc = ldc(fy, fc, j[0][5])
            else:
                print("Columna para zonas laterales\n")
                print("Para unión superior\n")
                ldc = ldc(fy, fc, j[0][5])
                print("Longitud de empalme unión viga-columna = ", ldc, "[cm]\n")
                lG = lGanchoC(j[0][5], fc, fy, j[0][1], dp)
                print("Longitud de gancho-remate = ", lG, "[cm]")
                print("Para unión inferior\n")
                ldc = ldc(fy, fc, j[0][5])

```

```

        print("Longitud de empalme unión viga-columna = ", ldc, "[cm]\n")
elif piso==1:
    if cont>npisos:
        print("Columna para zonas centrales\n")
        print("Para unión superior\n")
        ldc = ldc(fy, fc, j[0][5])
        print("Longitud de empalme unión viga-columna = ", ldc, "[cm]\n")
    else:
        print("Columna para zonas laterales\n")
        print("Para unión superior\n")
        ldc = ldc(fy, fc, j[0][5])
        print("Longitud de empalme unión viga-columna = ", ldc, "[cm]\n")
        lG = lGanchoC(j[0][5], fc, fy, j[0][1], dp)
        print("Longitud de gancho-remate = ", lG, "[cm]")
    else:
        if cont>npisos:
            print("Columna para zonas centrales\n")
            print("Para unión superior\n")
            lG = lGanchoC(j[0][5], fc, fy, j[0][1], dp)
            print("Longitud de gancho-remate = ", lG, "[cm]")
            print("Para unión inferior\n")
            lG = lGanchoC(j[0][5], fc, fy, j[0][1], dp)
            print("Longitud de gancho-remate = ", lG, "[cm]")
        else:
            print("Columna para zonas laterales\n")
            print("Para unión superior\n")
            lG = lGanchoC(j[0][5], fc, fy, j[0][1], dp)
            print("Longitud de gancho-remate = ", lG, "[cm]")
            print("Para unión inferior\n")
            ldc = ldc(fy, fc, j[0][5])
            print("Longitud de empalme unión viga-columna = ", ldc, "[cm]\n")

# lista1 --> [costo0, n° ramas1, de_externo2, espaciamento3, de_interno4, n° estribos5,
largo16, largos27, largo_tot28, d_ramas9, dist10]
# lista2 --> [costo, n° ramas, de_externo, espaciamento, de_interno, n° estribos, largo1,
largos2, largo_tot2, d_ramas, dist]
# lista3 --> [costo, n° ramas, de_externo, espaciamento, de_interno, n° estribos, largo1,
largos2, largo_tot2, d_ramas, dist]

"""Refuerzo transversal"""
# return [lista1, lista2, lista3, costo_total, vu1, vu2, hvig]

print("\n\nRefuerzo transversal\n")

print("\nZonas de rótula plástica y nodo\n")

print("\nUbicación RP: de 0 -", j[1][0][10], "[cm] y ", j[0][20]*100-j[1][0][10]-j[1][6], "-
", j[0][20]*100-j[1][6], "[cm]:")
print("\nUbicación RP: de", j[0][20]*100-j[1][6], "-", j[0][20]*100, "[cm]:")
print("N° ramas : ", j[1][0][1])
print("N° de estribos rótulas : ", j[1][0][5])
print("Espaciamento : ", j[1][0][3], "[cm]")
est1=int(j[1][6] / j[1][0][5]) + 1+j[1][0][5]
print("N° Estribos nodo : ", int(j[1][6]/j[1][0][5])+1)
print("\nRefuerzo exterior\n")
print("Diámetro estribo exterior: ", j[1][0][2], "[cm]")
print("Largo del estribo exterior", j[1][0][6], "[cm]")
print("Ubicación entre ejes de barras horizontales: x =", j[1][0][9][0][0], "[cm] y x
=", j[1][0][9][0][1], "[cm]")
print("Ubicación entre ejes de barras verticales: y =", j[1][0][9][0][0], "[cm] e y
=", j[1][0][9][0][1], "[cm]")
if j[1][0][1]>2:
    print("\nRefuerzo interior\n")
    if j[1][0][1]%2!=0:
        print("Diámetro de estribos y trabas interiores", j[1][0][4], "[cm]")
        if len(j[1][0][7])-1>0:
            for i in range(len(j[1][0][7])-1):
                print("Largo estribo interior n°", i+1, "=", j[1][0][7][i], "[cm]")
                print("Ubicación entre ejes de barras horizontales: x =",
j[1][0][9][i+1][0], "[cm] y x =",
j[1][0][9][i+1][1], "[cm]")
                print("Ubicación entre ejes de barras verticales: y =", j[1][0][9][0][0],
"[cm] e y =",
j[1][0][9][0][1], "[cm]")
            print("Largo de traba interior n°1 =", j[1][0][7][-1], "[cm]")
            #Revisar
            print("Ubicación entre ejes de barras horizontales: x =", j[1][0][9][-1][0],
"[cm]")
            print("Ubicación entre ejes de barras verticales: y =", j[1][0][9][0][0], "[cm] e
y =",
j[1][0][9][0][1], "[cm]")
    else:

```

```

        print("Diámetro de estribos interiores",j[1][0][4],"[cm]")
        if len(j[1][0][7])-1>0:
            for i in range(len(j[1][0][7])-1):
                print("Largo estribo interior n°",i+1,"=",j[1][0][7][i],"[cm]")
                print("Ubicación entre ejes de barras horizontales: x =", j[1][0][9][i] +
1[0], "[cm] y x =",
                    j[1][0][9][i + 1][1], "[cm]")
                print("Ubicación entre ejes de barras verticales: y =", j[1][0][9][0][0],
"[cm] e y =",
                    j[1][0][9][0][1], "[cm]")
            print("Largo de estribo interior n°", len(j[1][0][7]), "=", j[1][0][7][-1],
"[cm]")
            print("Ubicación entre ejes de barras horizontales: x =", j[1][0][9][-1][0], "[cm]
y x =",
                    j[1][0][9][-1][1], "[cm]")
            print("Ubicación entre ejes de barras verticales: y =", j[1][0][9][0][0], "[cm] e
y =",
                    j[1][0][9][0][1], "[cm]")

    print("\n\nEmpalme central\n")

    print("Ubicación : de",j[1][0][10],"-",
        j[0][20]*100-j[1][0][10]-j[1][6],"[cm]")
    distancia=j[0][20]*100-j[1][0][10]-j[1][6]-j[1][0][10]

    print("N° ramas : ", j[1][2][1])
    print("N° de estribos : ",int(distancia/j[1][2][3]))
    est2=int(distancia/j[1][2][3])
    print("Espaciamiento : ", j[1][2][3], "[cm]")
    print("\nRefuerzo exterior\n")
    print("Diámetro estribo exterior: ", j[1][2][2], "[cm]")
    print("Largo del estribo exterior", j[1][2][6], "[cm]")
    print("Ubicación entre ejes de barras horizontales: x =", j[1][2][9][0][0], "[cm] y x =",
j[1][2][9][0][1],
        "[cm]")
    print("Ubicación entre ejes de barras verticales: y =", j[1][2][9][0][0], "[cm] e y =",
j[1][2][9][0][1],
        "[cm]")
    if j[1][2][1] > 2:
        print("\nRefuerzo interior\n")
        if j[1][2][1] % 2 != 0:
            print("Diámetro de estribos y trabas interiores", j[1][2][4], "[cm]")
            if len(j[1][2][7]) - 1 > 0:
                for i in range(len(j[1][2][7]) - 1):
                    print("\nLargo estribo interior n°", i + 1, "=", j[1][2][7][i], "[cm]")
                    print("Ubicación entre ejes de barras horizontales: x =",
j[1][2][9][i+1][0], "[cm] y x =",
                            j[1][2][9][i + 1][1], "[cm]")
                    print("Ubicación entre ejes de barras verticales: y =", j[1][2][9][0][0],
"[cm] e y =",
                            j[1][2][9][0][1], "[cm]")
                print("\nLargo de traba interior n°1 =", j[1][2][7][-1], "[cm]")
                print("Ubicación entre ejes de barras horizontales: x =", j[1][0][9][-1][0],
"[cm]")
                print("Ubicación entre ejes de barras verticales: y =", j[1][2][9][0][0], "[cm] e
y =",
                            j[1][2][9][0][1], "[cm]")
            vol1 = (j[1][0][6] * acir(j[1][0][2]) + sum(j[1][0][7])*2 * acir(j[1][0][4]))*est2
            vol2 = (j[1][2][6] * acir(j[1][2][2]) + sum(j[1][2][7])*2 * acir(j[1][2][4]))*est2
            print("Peso total de estribos : ", round((vol1+vol2)*0.007850,1),"[kg]")
            print("Peso total barras longitudinales : ",
round(sum(j[0][13])*(j[1][2][10]+j[0][20])*0.007850,1),"[kg]")
            acerT=round(((vol1+vol2)+sum(j[0][13])*(j[1][2][10]+j[0][20]*100))*0.00785,1)
            print("Volumen Hormigón : ", j[0][20]*j[0][1]*j[0][2]*0.0001,"[m3]")
            print("Costo acero : $",round(acerT*1000,1))
            print("Costo hormigón : $",j[0][20]*j[0][1]*j[0][2]*7.5)
            costoT=round(acerT*1000,1)+j[0][20]*j[0][1]*j[0][2]*7.5
            print("Costo total de columna tipo:
$,round(acerT*1000,1)+j[0][20]*j[0][1]*j[0][2]*7.5)

    else:
        print("Diámetro de estribos interiores", j[1][2][4], "[cm]")
        if len(j[1][2][7]) - 1 > 0:
            for i in range(len(j[1][2][7]) - 1):
                print("\nLargo estribo interior n°", i + 1, "=", j[1][2][7][i], "[cm]")
                print("Ubicación entre ejes de barras horizontales: x =",
j[1][2][9][i+1][0], "[cm] y x =",
                        j[1][2][9][i+1][1], "[cm]")
                print("Ubicación entre ejes de barras verticales: y =", j[1][2][9][0][0],
"[cm] e y =",
                        j[1][2][9][0][1], "[cm]")
            print("\nLargo de estribo interior n°", len(j[1][2][7]), "=", j[1][2][7][-1],
"[cm]")
            print("Ubicación entre ejes de barras horizontales: x =", j[1][0][9][-1][0],
"[cm]")

```

```

y =",
    print("Ubicación entre ejes de barras verticales: y =", j[1][2][9][0][0], "[cm] e
        j[1][2][9][0][1], "[cm]")

    vol1 = (j[1][0][6] * acir(j[1][0][2]) + sum(j[1][0][7])*2 * acir(j[1][0][4]))*est2
    vol2 = (j[1][2][6] * acir(j[1][2][2]) + sum(j[1][2][7])*2 * acir(j[1][2][4]))*est2
    print("Peso total de estribos : ", round((vol1+vol2)*0.007850,1), "[kg]")
    print("Peso total de barras longitudinales : ",
round(sum(j[0][13]))*(j[1][2][10]+j[0][20])*0.007850,1), "[kg]")
    acerT=round((vol1+vol2)+sum(j[0][13]))*(j[1][2][10]+j[0][20]*100)*0.00785,1)
    print("Volumen Hormigón : ", j[0][20]*j[0][1]*j[0][2]*0.0001, "[m3]")
    print("Costo acero : $", round(acerT*1000,1))
    print("Costo hormigón : $", j[0][20]*j[0][1]*j[0][2]*7.5)
    costoT=round(acerT*1000,1)+j[0][20]*j[0][1]*j[0][2]*7.5
    print("Costo total de columna tipo:
$, round(acerT*1000,1)+j[0][20]*j[0][1]*j[0][2]*7.5)

    """Resultados"""

    print("\n\nResultados\n")
    print("Flexión\n")
    print("Mayor excentricidad", round(j[0][22] * 100,1), "cm\n")
    print("Momentos\n")
    print("Mu_max = ", j[0][17], "[tf-m]")
    print("Momento nominal ajustado a Mu y Pu máximos")
    print("ØMn1 = ", j[0][15], "[tf-m]")
    print("Momento nominal ajustado a Mu máximo debido a mayor excentricidad:")
    print("ØMn2 = ", j[0][23], "[tf-m]\n")

    print("Cargas\n")
    print("Pu_max = ", j[0][18], "[tf]")
    print("Pu_min = ", j[0][19], "[tf]")
    print("Carga nominal que verifica Pu_max:")
    print("ØPn1 = ", j[0][16], "[tf]")
    print("Carga nominal que verifica Pu_min:")
    print("ØPn2 = ", j[0][24], "[tf]\n")
    print("F.U. 1 = ", j[0][7], "%")
    print("F.U. 2 = ", j[0][8], "%\n")

    print("Corte")

    print("\nCorte en zona de rótula plástica")
    phiVn1 = round((2*acir(j[1][0][2])+acir(j[1][0][4]))*(j[1][0][1]-2))*fy*(j[0][1]-
j[0][27])/j[1][0][3],1)
    print("ØVn1 = ", round(phiVn1/1000,1), "[tf]")
    fuV1 = round(100*j[1][4]/(phiVn1),1)
    print("F.U.1 = ", fuV1, "%\n")

    # print("Corte en zona central")
    # phiVn2 = round((2*acir(j[1][1][2])+acir(j[1][1][4]))*(j[1][1][1]-2))*fy*(j[0][1]-
j[0][27])/j[1][1][3],1)
    # print("ØVn2 = ", round(phiVn2/1000,1), "[tf]")
    # fuV2 = round(100*j[1][5]/(phiVn2),1)
    # print("F.U.2 = ", fuV2, "%\n")

    print("Corte en zona de empalme")
    phiVn3 = round((2*acir(j[1][2][2])+acir(j[1][2][4]))*(j[1][2][1]-2))*fy*(j[0][1]-
j[0][27])/j[1][2][3],1)
    print("ØVn2 = ", round(phiVn3/1000,1), "[tf]")
    fuV3 = round(100*j[1][5]/(phiVn3),1)
    print("F.U.2 = ", fuV3, "%\n")
    print("\n")
    tempa=costoT*2 if tipo==1 else costoT*(nbahias-1)
    acum+=tempa
    return acum

def max_ind(lista,ind):
    temp=[]
    for i in range(len(lista)):
        maxim=0
        for j in range(len(lista[0])):
            if lista[i][j][ind]>maxim:
                maxim=lista[i][j][ind]
                list1=lista[i][j]
        temp.append(list1)
    return temp

def optimusFrame(tabla, largosC, largosV, dimV, ch, cs, b1, dp, es, ey, eu, fc, fy, dList,
deList, hColMax, hColMin):
    ai=2.26
    dList=[16,18,22,25,28,32,36]
    deList=[10,12]
    combis = 7
    combi_e = 4
    combi_s = 3

```

```

tab = tabla
filtro=filtroCV(combis, combi_e, combi_s, tab, largosV, largosC)
listav=filtro[0]
listac=filtro[1]
exc_col=filtro[2]
mpp1=[max([max([max(listav[i][j][0][2], listav[i][j][1][2]) for j in range(len(listav[0]))))
        for k in range(len(listav[0][0]))]) for i in range(len(listav))]
mpp2=[mpp1[j] for i in range(len(listav[0])) for j in range(len(listav))]
mpp3=[max(listav[i][0][0][2], listav[i][-1][1][2]) for i in range(len(listav))]
mnn1=[min([min([min(listav[i][j][0][3], listav[i][j][1][3]) for j in range(len(listav[0]))))
        for k in range(len(listav[0][0]))]) for i in range(len(listav))]
mnn2=[mnn1[j] for i in range(len(listav[0])) for j in range(len(listav))]
mnn3 = [max(listav[i][0][0][3], listav[i][-1][1][3]) for i in range(len(listav))]
allVuL = [[listav[i][j][0][6], listav[i][j][0][7], listav[i][j][1][6],
        listav[i][j][1][7]] for j in range(len(listav[0])) for i in range(len(listav))]
wo1 = [max([max(listav[i][j][0][4], listav[i][j][1][4]) for j in range(len(listav[0]))]) for i in
range(len(listav))]
wo2 = [wo1[j] for i in range(len(listav[0])) for j in range(len(listav))]
minLo = [min(i) for i in largosV]
maxLo = [max(i) for i in largosV]
lv = []
for i in allVuL:
    a=[[], [], [], []]
    a[0].append(max([i[j][0][0] for j in range(len(i))]))
    a[0].append(max([i[j][0][1] for j in range(len(i))]))
    a[1].append(max([i[j][1][0] for j in range(len(i))]))
    a[1].append(max([i[j][1][1] for j in range(len(i))]))
    a[1].append(min([i[j][1][2] for j in range(len(i))]))
    a[1].append(max([i[j][1][3] for j in range(len(i))]))
    a[2].append(min([i[j][2][0] for j in range(len(i))]))
    a[2].append(min([i[j][2][1] for j in range(len(i))]))
    a[3].append(min([i[j][3][0] for j in range(len(i))]))
    a[3].append(min([i[j][3][1] for j in range(len(i))]))
    a[3].append(min([i[j][3][2] for j in range(len(i))]))
    a[3].append(max([i[j][3][3] for j in range(len(i))]))
    lv.append(a)

lv2=[[i for j in range(len(allVuL[0]))] for i in lv]
listavig = [[mpp2[i][j], mnn2[i][j], wo2[i][j], largosV[i][j], lv2[i][j], dimV[i][j]]
        for j in range(len(listav[0])) for i in range(len(listav))]
listavig2 = [[listavig[i][0] for i in range(len(listavig))]
detvig2=matElemV(listavig2, cH, cS, b1, dp, es, ey, eu, fc, fy, dList, ai, deList, 5, nbahias,
dimC)

detvig = [[detvig2[j] for i in range(len(listavig[0])) for j in range(len(listavig))]
listacol = [[max(abs(listac[i][j][0][k]), abs(listac[i][j][1][k])) for k in range(6))
        for j in range(len(listac[0])) for i in range(len(listac))]
exc_col=[exc_col[i][j][0] for j in range(len(exc_col[0])) for i in range(len(exc_col))]
exc1=max_ind([exc_col[i][0], exc_col[i][-1]] for i in range(len(exc_col))), 2)
exc2 = max_ind([exc_col[i][1:-1] for i in range(len(exc_col))], 2)
tempcol = extMat(listacol, 4)
tempvig = [[abs(listavig[i][j][0]), abs(listavig[i][j][1])]
        for j in range(len(listavig[0])) for i in range(len(listavig))]
colDef=repMat(listacol, critVC(tempvig, tempcol), 4)

lc1 = []
for i in range(len(colDef)):
    col1=[max(colDef[i][0][j], colDef[i][-1][j]) for j in range(len(colDef[0][0]))]+exc1[i]
    col2=[max([colDef[i][k][j] for k in range(len(colDef[0]-2))] for j in
range(len(colDef[0][0]))]+exc2[i]
    lc1.append([col1, col2])
detcol=[]
cont=0
listC_bh1 = []
listC_bh2 = []
hmax1 = dimC[0][0][0]
hmax2 = dimC[0][1][0]
hmin1 = dimC[0][0][2]
hmin2 = dimC[0][1][2]
for j in range(len(lc1[0])):
    tempC=[]
    for i in range(len(lc1)):
        if j==0:
            cont+=1
            elem=optimuscol(b1, dp, es, eu, ey, fc, fy, lc1[i][j][4], round(lc1[i][j][7]/1000,1),
                    round(lc1[i][j][6]/1000,1), lc1[i][j][0], dList, hmax1, hmin1, cH,
                    cs, lc1[i][j][5], lc1[i][j][2], lc1[i][j][3], deList, 1,
dimV[i][0][0]-5)
            titulo = str("columna tipo "+ str(j+1)+ " del piso " + str(i+1))
            xyplotCurv(elem[0][13], elem[0][2], elem[0][1], dp, eu, fy, fc, b1, es, ey,
            elem[0][14], elem[0][10], lc1[i][j][4], lc1[i][j][0], elem[0][15], elem[0][16],
titulo)

            # optimuscol(b1, dp, es, eu, ey, fc, fy, muC, muCmin, puCmin, puCmax, dList, hmax,
hmin, cH, cS, H, vu,
            #
            vue, deList, iguales)

```

```

        # optimo = [minor0, h1, b2, j3, k4, l5, m6, fu7, fu2 8, cuan9, cF[0]10, cF2[0]11, e12,
        # alist13, ylist14, cF[1]15, cF[2]16, muC17, puCmax18, puCmin19, H20, iguales21,
        round(muCmin / puCmin, 3)22,
        # cF2[1]23, cF2[2]24, costo1 25, costo2 26, dp27]
        tempC.append(elem)
        hmax1=elem[0][1]
        hmin1=hmax1-5
        listC_bh1.append([elem[0][2],elem[0][1]])
    else:
        cont+=1
        elem = optimusCol(b1, dp, es, eu, ey, fc, fy, lC1[i][j][4], round(lC1[i][j][7] / 1000,
1),
                                round(lC1[i][j][6] / 1000, 1), lC1[i][j][0], dList, hmax2, hmin2,
CH, CS,
                                lC1[i][j][5], lC1[i][j][2], lC1[i][j][3], deList, 1,dimV[i][1][0]-
5)
        # optimusCol(b1, dp, es, eu, ey, fc, fy, muC, muCmin, puCmin, puCmax, dList, hmax,
hmin, CH, CS, H, vu,
        #
                                vue, deList, iguales)

        titulo = str("Columna tipo "+ str(j+1)+ " del piso " + str(i+1))
        XYplotCurv(elem[0][13], elem[0][2], elem[0][1], dp, eu, fy, fc, b1, es, ey,
elem[0][14], elem[0][10],
                                lC1[i][j][4], lC1[i][j][0], elem[0][15], elem[0][16], titulo)
        hmax2=elem[0][1]
        hmin2=hmax2-5
        listC_bh2.append([elem[0][2],elem[0][1]])
        tempC.append(elem)
        detcol.append(tempC)
        listC_bh=[]
        cont=0
        for i in range(len(listaC)):
            for j in range(len(listaC[0])):
                if j==0 or j==len(listaC):
                    cont+=1
                    listC_bh.append([listC_bh1[i][0],listC_bh1[i][1]])
                else:
                    cont += 1
                    listC_bh.append([listC_bh1[i][0],listC_bh1[i][1]])
        listV_bh=[]
        cont=0
        for i in detvig:
            for j in i:
                cont+=1
                listV_bh.append((j[0][0][2],j[0][0][1]))
        # print(detcol)
        detC=detCol(detcol)
        # print(detvig2)
        detV=detVig(detvig2,nbahias,listV_bh)
        # asdf1=[detcol[i][j][0][0] for j in range(detcol[0]) for i in range(detcol)]

        col1=sum([detcol[0][i][0][0]*2 for i in range(len(detcol[0]))])
        col2=sum([detcol[1][i][0][0]*(nbahias-1) for i in range(len(detcol[0]))])
        asdf1=col1+col2
        asdf2=sum([detvig2[i][j][0][0] for i in range(len(detvig2)) for j in range(len(detvig2[0]))])
        print("valor columnas", asdf1)
        print("valor acero", asdf2)
        print("valor total", asdf1+asdf2)
        print("drift maximo :",p.tabla['drift'])

        # print(detcol[0][0][0][0])
        # print("precio columnas $",asdf1)
        # print("precio vigas $",asdf2)
        # print("La estructura cuesta en total : $",asdf1+asdf2,"\\n\\n")
        list_bh = listC_bh+listV_bh

    return [detcol,detvig2, list_bh]

from time import time
t1=time()
asd=optimusFrame(tabla, largosC, largosV, dimV, CH, CS, b1, dp,
                es, ey, eu, fc, fy, dList, deList, hColMax, hColMin)

t2=time()-t1
# print("tiempo de ejecución",round(t2,5),"segundos")
# print(asd[0], "\\n",asd[1])
print(asd[2])
optijandro()

```

