UNIVERSIDAD DE HUÁNUCO

FACULTAD DE INGENIERA

PROGRAMA ACADÉMICO DE INGENIERA CIVIL

TESIS

“Desarrollo de OpenRSE en Lua para mejorar la eficiencia y accesibilidad en el análisis estructural bidimensional de estructuras hiperestáticas, Huánuco, 2024”

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO CIVIL

AUTOR: Rosas Placido, Alexander

ASESOR: Narro Jara, Luis Fernando

HUÁNUCO – PERÚ

2025

U

D

H

TIPO DEL TRABAJO DE INVESTIGACIÓN:

- Tesis (X)
- Trabajo de Suficiencia Profesional ( )
- Trabajo de investigación ( )
- Trabajo Académico ( )

LÍNEAS DE INVESTIGACIÓN: Estructuras
AÑO DE LA LÍNEA DE INVESTIGACIÓN (2020)

CAMPO DE CONOCIMIENTO OCDE:
Área: Ingeniería, Tecnología
Sub área: Ingeniería Civil
Diciplina: Ingeniería Civil

DATOS DEL PROGRAMA:

Nombre del Grado/Título a recibir: Titulo Profesional de Ingeniero Civil
Código del Programa: P07
Tipo de Financiamiento:

- Propio (X)
- UDH ( )
- Fondos Concursables ( )

DATOS DEL AUTOR:
Documento Nacional de Identidad (DNI): 72751423

DATOS DEL ASESOR:
Documento Nacional de Identidad (DNI): 18206328
Grado/Título: Maestro en ingeniería con mención en gestión ambiental y desarrollo sostenible
Código ORCID: 0000-0003-4008-7633

DATOS DE LOS JURADOS:

N°	APELLIDOS Y NOMBRES	GRADO	DNI	Código ORCID
1	Arteaga Espinoza, Ingrid Delia Dignarda	Máster en dirección de proyectos	73645168	0009-0001-0745-5433
2	Barboza Quispe, Juan Carlos	Magister en educación mención en docencia y gestión educativa	41541171	0000-0002-4070-3830
3	Miraval Rojas, Biseth	Maestro en gestión y negocios, con mención en gestión de proyectos	47474699	0000-0001-5605-3003

UNIVERSIDAD DE HUANUCO

Facultad de Ingeniería

PROGRAMA ACADÉMICO DE INGENIERÍA CIVIL

ACTA DE SUSTENTACIÓN DE TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO(A) CIVIL

En la ciudad de Huánuco, siendo las 18:30 horas del día **lunes 06 de octubre de 2025**, en cumplimiento de lo señalado en el Reglamento de Grados y Títulos de la Universidad de Huánuco, se reunieron los **Jurados Calificadores** integrado por los docentes:

- MG. INGRID DELIA DIGNARDA ARTEAGA ESPINOZA — PRESIDENTE
- MG. JUAN CARLOS BARBOZA QUISPE — SECRETARIO
- MG. BISETH MIRAVAL ROJAS — VOCAL

Nombrados mediante la RESOLUCIÓN Nº 2040-2025-D-FI-UDH, para evaluar la **Tesis** intitulada: **"DESARROLLO DE OPENRSE EN LUA PARA MEJORAR LA EFICIENCIA Y ACCESIBILIDAD EN EL ANÁLISIS ESTRUCTURAL BIDIMENSIONAL DE ESTRUCTURAS HIPERESTÁTICAS, HUÁNUCO, 2024"**, presentado por el (la) Bachiller. Bach. **Alexander ROSAS PLACIDO**, para optar el Título Profesional de Ingeniero(a) Civil.

Dicho acto de sustentación se desarrolló en dos etapas: exposición y absolución de preguntas: procediéndose luego a la evaluación por parte de los miembros del Jurado.

Habiendo absuelto las objeciones que le fueron formuladas por los miembros del Jurado y de conformidad con las respectivas disposiciones reglamentarias, procedieron a deliberar y calificar, declarándolo(a) ...APROBADO..... por ...UNANIMIDAD..... con el calificativo cuantitativo de ...14..... y cualitativo de SUFICIENTE..... (Art. 47).

Siendo las 19:30.. horas del día **06 del mes de octubre** del año 2025, los miembros del Jurado Calificador firman la presente Acta en señal de conformidad.

MG. INGRID DELIA DIGNARDA ARTEAGA ESPINOZA
DNI: 73645168
ORCID: 0009-0001-0745-5433
PRESIDENTE

MG. JUAN CARLOS BARBOZA QUISPE
DNI: 41541171
ORCID: 0000-0002-4070-3830
SECRETARIO (A)

MG. BISETH MIRAVAL ROJAS
DNI: 47474699
ORCID: 0000-0001-5605-3003
VOCAL

VICERRECTORADO DE INVESTIGACIÓN

UNIVERSIDAD DE HUÁNUCO

CONSTANCIA DE ORIGINALIDAD

El comité de integridad científica, realizó la revisión del trabajo de investigación del estudiante: ALEXANDER ROSAS PLACIDO, de la investigación titulada "DESARROLLO DE OPENRSE EN LUA PARA MEJORAR LA EFICIENCIA Y ACCESIBILIDAD EN EL ANÁLISIS ESTRUCTURAL BIDIMENSIONAL DE ESTRUCTURAS HIPERESTÁTICAS, HUÁNUCO, 2024", con asesor(a) LUIS FERNANDO NARRO JARA, designado(a) mediante documento: RESOLUCIÓN N° 1368-2024-D-FI-UDH del P. A. de INGENIERÍA CIVIL.

Puede constar que la misma tiene un índice de similitud del 7 % verificable en el reporte final del análisis de originalidad mediante el Software Turnitin.

Por lo que concluyo que cada una de las coincidencias detectadas no constituyen plagio y cumple con todas las normas de la Universidad de Huánuco.

Se expide la presente, a solicitud del interesado para los fines que estime conveniente.

Huánuco, 26 de agosto de 2025

UNIVERSIDAD DE HUÁNUCO
RESPONSABLE DE INTEGRIDAD CIENTÍFICA
HUÁNUCO - PERÚ

RICHARD J. SOLIS TOLEDO
D.N.I.: 47074047
cod. ORCID: 0000-0002-7629-6421

UNIVERSIDAD DE HUÁNUCO
RESPONSABLE DE TURNITIN
HUÁNUCO - PERÚ

MANUEL E. ALIAGA VIDURIZAGA
D.N.I.: 71345687
cod. ORCID: 0009-0004-1375-5004

173. Rosas Placido, Alexander.docx

INFORME DE ORIGINALIDAD

7%	7%	2%	2%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	vsip.info Fuente de Internet	1%
2	ingmec.ual.es Fuente de Internet	<1%
3	hdl.handle.net Fuente de Internet	<1%
4	memoriascimted.com Fuente de Internet	<1%
5	moam.info Fuente de Internet	<1%
6	kupdf.net Fuente de Internet	<1%

UNIVERSIDAD DE HUÁNUCO
RESPONSABLE DE INTEGRIDAD CIENTÍFICA
HUÁNUCO - PERÚ

RICHARD J. SOLIS TOLEDO
D.N.I.: 47074047
cod. ORCID: 0000-0002-7629-6421

UNIVERSIDAD DE HUÁNUCO
RESPONSABLE DE TURNITIN
HUÁNUCO - PERÚ

MANUEL E. ALIAGA VIDURIZAGA
D.N.I.: 71345687
cod. ORCID: 0009-0004-1375-5004

DEDICATORIA

A mis padres, por ser el pilar fundamental en cada etapa de mi formación, por enseñarme con el ejemplo el valor del esfuerzo, la constancia y la honestidad. Gracias por su paciencia infinita en los momentos difíciles, por su apoyo silencioso pero firme, y por brindarme siempre un espacio de confianza, motivación y afecto. Esta tesis es también el fruto de su entrega y dedicación, y a ustedes les pertenece tanto como a mí.

AGRADECIMIENTOS

Expreso mi más sincero agradecimiento a todas las personas que hicieron posible la culminación de esta tesis. A mis padres, por ser una presencia firme y alentadora a lo largo de todo este proceso, por comprender mis ausencias y por sostenerme con su apoyo moral y emocional en cada etapa. A mi asesor, por acompañar este trabajo durante su desarrollo y por formar parte del proceso académico que lo hizo posible. A la Universidad de Huánuco, por brindarme los medios formativos y los recursos necesarios para completar mi formación profesional. Asimismo, extiendo mi reconocimiento a los docentes que contribuyeron a mi crecimiento académico a lo largo de la carrera, y a todas las personas e instituciones que, de forma directa o indirecta, colaboraron en la realización de este proyecto.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

La presente investigación tiene como objetivo principal desarrollar y validar el software OpenRSE, programado en el lenguaje Lua, como una herramienta de análisis estructural bidimensional orientada a mejorar la eficiencia y accesibilidad en el estudio de estructuras hiperestáticas. El proyecto surge como respuesta a las limitaciones de accesibilidad, altos costos de licencias y falta de transparencia en los programas comerciales actuales como SAP2000 y Robot Structural.

OpenRSE fue desarrollado en el entorno de programación de la plataforma TI-Nspire CX CAS, empleando una arquitectura modular basada en el método matricial de rigidez. El software destaca por su diseño eficiente, su interfaz gráfica clara y su portabilidad, siendo ejecutable en múltiples sistemas operativos sin necesidad de equipos de alto rendimiento. Además, permite la visualización detallada y paso a paso de los procesos de cálculo estructural, lo que contribuye a una mayor comprensión y control del análisis.

Para validar su precisión, se analizaron diez modelos estructurales diversos, comparando los resultados obtenidos con OpenRSE frente a los softwares SAP2000 y Robot Structural. Se calcularon errores absolutos y errores relativos porcentuales, evidenciando una alta concordancia con los resultados de referencia y variaciones mínimas en los parámetros evaluados. Asimismo, se evaluó su eficiencia operativa en términos de tiempo de procesamiento, compatibilidad, tipo de licencia y tamaño del archivo de instalación, posicionando a OpenRSE como una alternativa ligera, precisa y de código abierto frente a las opciones comerciales.

Los hallazgos respaldan la hipótesis general y específicas planteadas, validando que OpenRSE mejora significativamente la accesibilidad y eficiencia del análisis estructural bidimensional, constituyéndose en una solución viable, gratuita y adaptable para estudiantes, profesionales e investigadores en ingeniería civil.

Palabras clave: OpenRSE, análisis estructural, estructuras hiperestáticas, software libre, eficiencia, accesibilidad.

ABSTRACT

The main objective of this research is the development and validation of the OpenRSE software, programmed in the Lua language, as a tool for two-dimensional structural analysis aimed at improving efficiency and accessibility in the study of hyperstatic structures. The project addresses key limitations in current commercial software, such as high licensing costs, restricted accessibility, and lack of transparency in programs like SAP2000 and Robot Structural.

OpenRSE was developed within the programming environment of the TI-Nspire CX CAS platform, utilizing a modular architecture based on the stiffness matrix method. The software stands out for its efficient design, intuitive graphical interface, and portability, being executable across multiple operating systems without requiring high-performance hardware. Furthermore, it allows for detailed, step-by-step visualization of the internal calculation processes, which enhances understanding and control of the structural analysis.

To verify its accuracy, ten different structural models were analyzed, and the results obtained with OpenRSE were compared with those from SAP2000 and Robot Structural. Absolute and relative percentage errors were calculated, revealing a high level of agreement with reference results and minimal variations in key parameters. Operational efficiency was also evaluated in terms of processing time, platform compatibility, licensing type, and installation file size, positioning OpenRSE as a lightweight, precise, and open-source alternative to commercial options.

The findings support both the general and specific hypotheses proposed, validating that OpenRSE significantly enhances the efficiency and accessibility of two-dimensional structural analysis, making it a viable, free, and adaptable solution for students, professionals, and researchers in civil engineering.

Keywords: OpenRSE, structural analysis, hyperstatic structures, open-source software, efficiency, accessibility.

INTRODUCCIÓN

El análisis estructural es una herramienta fundamental en la ingeniería civil, utilizada para prever el comportamiento de estructuras bajo diversas condiciones de carga. En los últimos años, el desarrollo de software especializado ha permitido optimizar este proceso, sin embargo, la mayoría de los programas existentes presentan altos costos, licencias restrictivas y escasa transparencia en su funcionamiento interno, lo cual limita su uso en entornos educativos o profesionales con recursos limitados. Además, la ausencia de opciones portables y de libre acceso dificulta el aprendizaje autónomo y la adaptación tecnológica a diferentes plataformas.

En respuesta a esta problemática, la presente investigación propone el desarrollo y validación de OpenRSE, un software de análisis estructural bidimensional enfocado en estructuras hiperestáticas, programado en Lua y desarrollado en el entorno de la plataforma TI-Nspire CX CAS. Su diseño modular, su arquitectura orientada a objetos y su compatibilidad con múltiples sistemas operativos permiten ofrecer una alternativa accesible, eficiente y de código abierto, que busca democratizar el uso de herramientas avanzadas en el campo de la ingeniería estructural.

La tesis se organiza en cinco capítulos:

Capítulo I presenta el planteamiento del problema, los objetivos de la investigación, así como sus justificaciones teóricas, práctica, metodológica, económica y social. También se describen las limitaciones y condiciones que aseguran la viabilidad del proyecto.

Capítulo II desarrolla el marco teórico, incluyendo antecedentes relevantes, fundamentos del análisis estructural y del método matricial de rigidez, así como las definiciones conceptuales, hipótesis y variables empleadas en la investigación.

Capítulo III expone la metodología utilizada, describiendo el enfoque cuantitativo, el diseño no experimental, la técnica de validación por comparación y el uso de una muestra dirigida de modelos estructurales para evaluar el desempeño de OpenRSE.

Capítulo IV presenta los resultados obtenidos, abordando el diseño del software, la lógica de cálculo implementada y la interfaz gráfica desarrollada. Se realiza una comparación detallada de resultados numéricos entre OpenRSE, SAP2000 y Robot Structural, evaluando tanto la precisión estructural como la eficiencia operativa del programa.

Capítulo V ofrece una discusión crítica de los resultados, valorando el cumplimiento de las hipótesis planteadas y analizando las fortalezas, limitaciones y proyecciones del software desarrollado en el ámbito profesional y académico.

Capítulo VI presenta las conclusiones que consolidan la validez del software OpenRSE como una herramienta útil y confiable, junto con recomendaciones orientadas a su mejora continua y aplicación futura.

CAPÍTULO I
PROBLEMA DE INVESTIGACIÓN

1.1. DESCRIPCIÓN DEL PROBLEMA

En el campo de la ingeniería estructural, el análisis bidimensional de estructuras es crucial para garantizar la seguridad y eficiencia de los proyectos. Sin embargo, existen barreras significativas en términos de accesibilidad y eficiencia. Los programas avanzados disponibles en la actualidad, como ETABS y SAP2000, tienen costos elevados, con licencias que varían entre $2,860 y $17,160, además de costos anuales de mantenimiento (Computers & Structure, Inc., 2023). Estos altos precios restringen el acceso a estas herramientas, lo que afecta directamente su accesibilidad para un amplio rango de usuarios.

Otro problema importante es la falta de acceso al código fuente. Tanto los programas privativos como los libres disponibles actualmente no ofrecen acceso a su código fuente, lo que impide su personalización o adaptación a necesidades específicas. Esto limita la flexibilidad y la posibilidad de mejorar las herramientas según las demandas del usuario, afectando la eficiencia en el desarrollo y aplicación de soluciones estructurales.

Además, estas herramientas suelen ser complejas y tienen una curva de aprendizaje prolongada, lo que dificulta su uso eficiente. Por otro lado, la falta de compatibilidad con dispositivos portátiles limita la capacidad de realizar análisis en el campo, una necesidad crucial para muchos proyectos.

El desarrollo de OpenRSE en LUA busca resolver estos problemas al proporcionar una herramienta de análisis estructural bidimensional que sea accesible, eficiente y completamente de código abierto. A diferencia de las opciones actuales, OpenRSE pondrá su código a disposición de los usuarios, permitiendo la personalización y mejora continua del software. Esto fomentará una mayor accesibilidad y colaboración, mejorando la eficiencia en el análisis

estructural, sin las barreras de costos o falta de flexibilidad que presentan las herramientas existentes. Al ser una solución portátil y fácil de usar, OpenRSE permitirá realizar análisis estructurales en diversas situaciones con mayor agilidad y precisión.

1.2. FORMULACIÓN DEL PROBLEMA

1.2.1. PROBLEMA GENERAL

¿De qué manera influye, en el análisis estructural bidimensional de estructuras hiperestáticas, el desarrollo de OpenRSE en Lua en la mejora de la eficiencia y accesibilidad, Huánuco, 2024?

1.2.2. PROBLEMAS ESPECÍFICOS

¿Qué características debe incluir el desarrollo de OpenRSE en Lua para mejorar la eficiencia y eficiencia en el análisis de estructuras hiperestáticas, Huánuco, 2024?

¿Qué metodologías de análisis estructural bidimensional de estructuras hiperestáticas deben integrarse en OpenRSE para garantizar un análisis preciso y eficiente, Huánuco, 2024?

¿Cómo se puede optimizar OpenRSE en Lua para mejorar la precisión y la velocidad en la verificación de estructuras hiperestáticas bidimensionales, Huánuco, 2024?

¿Cómo validar los resultados de OpenRSE comparándolos con los obtenidos por SAP2000 y Robot Structural, para garantizar su robustez y precisión en el análisis de estructuras hiperestáticas, Huánuco, 2024?

¿En qué medida el desarrollo de OpenRSE en Lua contribuye a mejorar la eficiencia y accesibilidad para los ingenieros en comparación con SAP2000 y Robot Structural, Huánuco, 2024?

1.3. OBJETIVO GENERAL

Determinar de qué manera influye el desarrollo de OpenRSE en Lua en la mejora de la eficiencia y accesibilidad en el análisis estructural bidimensional de estructuras hiperestáticas, Huánuco, 2024.

1.4. OBJETIVOS ESPECÍFICOS

Identificar las características que debe incluir el desarrollo de OpenRSE en Lua para mejorar la eficiencia y eficiencia en el análisis de estructuras hiperestáticas, Huánuco, 2024.

Integrar metodologías de análisis estructural bidimensional de estructuras hiperestáticas en OpenRSE para garantizar un análisis preciso y eficiente, Huánuco, 2024.

Optimizar OpenRSE en Lua para mejorar la precisión y la velocidad en la verificación de estructuras hiperestáticas bidimensionales, Huánuco, 2024.

Validar los resultados de OpenRSE comparándolos con los obtenidos por SAP2000 y Robot Structural, para garantizar su robustez y precisión en el análisis de estructuras hiperestáticas, Huánuco, 2024.

Evaluar en qué medida el desarrollo de OpenRSE en Lua contribuye a mejorar la eficiencia y accesibilidad para los ingenieros en comparación con SAP2000 y Robot Structural, Huánuco, 2024.

1.5. JUSTIFICACIÓN DE LA INVESTIGACIÓN

1.5.1. JUSTIFICACIÓN TEÓRICA

La investigación teórica en torno al desarrollo del software OpenRSE es crucial porque amplía el conocimiento sobre el diseño de software de análisis estructural accesible y eficiente. Al integrar

metodologías avanzadas de análisis estructural en una plataforma portátil y accesible, esta investigación facilita la exploración de nuevas formas de aplicar análisis estructural. Además, permite la validación de teorías de usabilidad y eficacia en el uso de tecnologías portátiles para el análisis estructural. Esta validación es fundamental para demostrar que el software de código abierto puede ser una solución viable y efectiva en diversas aplicaciones. Al abordar cómo las características del software mejoran la accesibilidad y eficiencia, esta investigación también contribuye al desarrollo teórico en el campo del diseño de software, proporcionando un marco para futuras investigaciones y desarrollos en esta área.

1.5.2. JUSTIFICACIÓN PRACTICA

La investigación sobre el desarrollo del software OpenRSE es esencial porque ofrece una solución accesible y eficiente para quienes necesitan herramientas económicas y fáciles de usar para el análisis estructural. OpenRSE mejora la portabilidad y la flexibilidad en el trabajo práctico, permitiendo realizar análisis estructurales en campo y facilitando verificaciones y ajustes en tiempo real en sitios de construcción. Su interfaz intuitiva y recursos didácticos simplifican el aprendizaje continuo, superando las barreras económicas y técnicas de las herramientas tradicionales. Al promover el uso de software de código abierto, OpenRSE se presenta como una alternativa viable y sostenible, mejorando la accesibilidad y eficiencia en el análisis estructural, y beneficiando a una amplia comunidad de usuarios en diversos contextos.

1.5.3. JUSTIFICACIÓN METODOLÓGICA

La investigación se centra en la necesidad de un enfoque práctico y detallado para desarrollar y evaluar el software OpenRSE. El uso de métodos de diseño y desarrollo iterativos es crucial porque permite la creación de prototipos y la realización de pruebas continuas, asegurando

que el software se adapte de manera efectiva a los requisitos definidos. Las comparaciones con métodos tradicionales y software de alto nivel, como SAP2000 y Robot Structural, son esenciales para demostrar mejoras en precisión y velocidad, validando así las ventajas del nuevo software. Además, las simulaciones en entornos controlados replican condiciones reales y facilitan la identificación de posibles problemas, lo que es fundamental para garantizar la efectividad del software. Basado en los resultados de estas comparaciones y simulaciones, se realizarán ajustes continuos al software, asegurando que OpenRSE cumpla con los altos estándares de calidad y funcionalidad requeridos. Este enfoque metodológico integral justifica la elección de métodos específicos y asegura que el software desarrollado mejore significativamente la accesibilidad y eficiencia del análisis estructural bidimensional.

1.5.4. JUSTIFICACIÓN ECONÓMICA

El desarrollo del software OpenRSE de código abierto se justifica económicamente debido a los significativos ahorros que ofrece en comparación con los altos costos de licencias y mantenimiento de los programas de análisis estructural tradicionales, como ETABS y SAP2000, cuyos precios pueden alcanzar hasta $17,160 con costos de mantenimiento anual de hasta $3,005 (CSI Structure, 2023). OpenRSE elimina estas barreras financieras, haciendo accesible una herramienta avanzada de análisis estructural sin necesidad de grandes inversiones. Además, permite la adaptación y mejora del software sin costos adicionales, posibilitando la reasignación de recursos hacia áreas críticas como equipos, formación y desarrollo de proyectos, promoviendo así el crecimiento y la competitividad del sector de la ingeniería civil en general.

1.5.5. JUSTIFICACIÓN SOCIAL

El desarrollo del software OpenRSE de código abierto tiene una fuerte justificación social, ya que promueve la igualdad de oportunidades

en el acceso a herramientas avanzadas de análisis estructural. Al eliminar los altos costos asociados con el software comercial, OpenRSE permite que una mayor cantidad de individuos, independientemente de su situación económica, accedan a tecnología de alta calidad. Esto es especialmente importante en países en desarrollo como Perú, donde los recursos educativos pueden ser limitados. Además, OpenRSE fomenta la colaboración y el intercambio de conocimientos en la comunidad de ingeniería civil, ya que el carácter de código abierto permite a los usuarios contribuir a la mejora y personalización del software según sus necesidades específicas. Esta accesibilidad y colaboración no solo mejoran la educación y la práctica, sino que también fortalecen las capacidades técnicas de la sociedad, contribuyendo al desarrollo sostenible y equitativo del sector de la ingeniería civil. En última instancia, OpenRSE ayuda a formar profesionales mejor preparados y más competitivos, lo que repercute positivamente en la infraestructura y el bienestar social general.

1.6. LIMITACIONES DE LA INVESTIGACIÓN

En el desarrollo de esta investigación se identificaron varias limitaciones que podrían representar obstáculos para alcanzar los objetivos planteados. La complejidad inherente al desarrollo del software OpenRSE en Lua es una de las principales. Contar con un nivel intermedio de conocimiento del lenguaje Lua puede afectar la eficiencia y efectividad en la implementación de las funcionalidades del software; un dominio más avanzado facilitaría la incorporación de características más sofisticadas y optimizadas.

Existe también la dificultad de crear una solución generalizada para el análisis estructural bidimensional de estructuras hiperestáticas. Muchas bibliografías y recursos académicos abordan el tema de manera fragmentada, enfocándose en secciones o casos específicos, lo que complica la unificación de los diversos enfoques y metodologías en un software cohesivo y funcional. Integrar diversas teorías y prácticas en una plataforma única y accesible añade un nivel adicional de complejidad al desarrollo de OpenRSE.

Durante la búsqueda de antecedentes y literatura relacionada, se presentaron obstáculos significativos. Gran parte de las investigaciones relevantes disponibles tienen más de cinco años de antigüedad y no reflejan los avances más recientes en el campo, limitando su utilidad como referencias actualizadas. Muchos repositorios y fuentes de información actuales se encuentran fuera de servicio o con acceso restringido, dificultando el acceso a investigaciones recientes y pertinentes. Esta escasez de referencias actualizadas limita la capacidad de establecer un marco teórico sólido y actualizado para la investigación.

Estas limitaciones podrían impactar en el alcance y profundidad de los resultados obtenidos. No obstante, mediante un enfoque metodológico riguroso y una planificación cuidadosa, es posible mitigar estos obstáculos y cumplir con los objetivos de la investigación. La colaboración con expertos en programación Lua y en análisis estructural, así como el uso de fuentes alternativas de información, podrían ser estrategias efectivas para superar las limitaciones identificadas.

1.7. VIABILIDAD DE LA INVESTIGACIÓN

La viabilidad de esta investigación se fundamenta en varios factores clave que aseguran su ejecución exitosa y la obtención de resultados relevantes. Se dispone de los recursos tecnológicos necesarios, incluyendo herramientas de desarrollo adecuadas para programar en Lua, lo que permite el desarrollo y prueba eficientes del software OpenRSE. Aunque el nivel de conocimiento del lenguaje es intermedio, se cuenta con las habilidades suficientes para llevar a cabo el desarrollo, y la amplia disponibilidad de recursos educativos en línea facilita la superación de cualquier desafío técnico que pueda surgir durante el proceso de programación.

Se tiene acceso a una extensa bibliografía sobre metodologías de análisis estructural bidimensional, lo cual proporciona una base sólida para el desarrollo teórico y práctico de OpenRSE. Esta literatura especializada aporta

los fundamentos necesarios para integrar las metodologías y teorías pertinentes en el software, asegurando su rigor y eficacia.

La clara definición de los objetivos y la identificación de las limitaciones previstas proporcionan una hoja de ruta detallada para el desarrollo del software. Esta planificación estratégica permite anticipar posibles obstáculos y establecer medidas para mitigarlos. La motivación por mejorar la accesibilidad y eficiencia en el análisis estructural bidimensional actúa como un impulso constante que orienta el esfuerzo hacia el cumplimiento de los objetivos propuestos.

En conjunto, estos factores demuestran que la investigación es viable y posee un alto potencial de éxito en el desarrollo de un software innovador y accesible. OpenRSE beneficiará a estudiantes y profesionales de la ingeniería civil al proporcionarles una herramienta eficaz y asequible para el análisis estructural. La combinación de recursos tecnológicos, conocimientos técnicos, acceso a bibliografía especializada y una planificación estructurada asegura que se podrán enfrentar y superar las dificultades que se presenten durante el proceso de investigación y desarrollo.

CAPÍTULO II
MARCO TEÓRICO

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

2.1.1. ANTECEDENTES INTERNACIONALES

De León León (2020), en su tesis titulada "Creación de software para la línea estructural de ingeniería civil", presentada en la Facultad de Ingenierías de la Universidad Cooperativa de Colombia en Santa Marta, Colombia, tuvo como objetivo principal desarrollar un software destinado a facilitar los cálculos estructurales relacionados con el centro de gravedad y el momento de inercia en el campo de la ingeniería civil. El proyecto se enfocó en crear una herramienta que automatizara estos cálculos para mejorar la eficiencia en las tareas diarias de ingenieros civiles. La investigación involucró el uso de la aplicación Smath Studio para programar una hoja de cálculo capaz de realizar los cálculos complejos relacionados con estos dos conceptos estructurales. La elección de Smath Studio se basó en su accesibilidad y en su capacidad para ser una herramienta de fácil manejo para los usuarios, permitiendo que estos ingresen los datos de entrada necesarios para obtener resultados inmediatos. Se realizaron varias etapas de pruebas y ajustes para asegurar que el software fuera capaz de realizar cálculos precisos, tanto para figuras geométricas simples como para figuras compuestas. Los resultados principales indicaron que el software desarrollado ofrece una forma eficiente y rápida de calcular el centro de gravedad y el momento de inercia, reduciendo considerablemente el tiempo necesario en comparación con los métodos tradicionales. Además, la herramienta resultó ser de fácil uso, lo que permite a los usuarios con distintos niveles de conocimiento técnico acceder a soluciones rápidas y precisas. Una de las ventajas clave del software es que puede ser utilizado tanto en la academia como en el campo profesional, permitiendo que ingenieros y estudiantes obtengan resultados inmediatos para diferentes tipos de

problemas estructurales. En conclusión, la tesis demostró que la creación de software personalizado, utilizando herramientas como Smath Studio, es una alternativa viable para simplificar los cálculos estructurales complejos, mejorando la accesibilidad y reduciendo los tiempos de ejecución, lo que impacta directamente en la productividad de los usuarios. La relevancia de esta tesis para la presente investigación radica en su enfoque hacia la creación de software accesible y eficiente en el área de la ingeniería estructural. El uso de Smath Studio como plataforma para desarrollar una herramienta de cálculo refleja una metodología práctica que respalda los objetivos de la investigación actual con OpenRSE, donde se busca crear un software que no solo facilite el análisis estructural bidimensional de estructuras hiperestáticas, sino que también mejore la accesibilidad y eficiencia en la obtención de resultados, ofreciendo una solución viable tanto para el ámbito académico como para la práctica profesional en ingeniería civil

Ramírez Vargas (2022), en su tesis titulada "Desarrollo de un programa de computador para el análisis lineal de estructuras aporticadas tridimensionales sometidas a cargas estáticas", presentada en el Departamento de Ingeniería Civil y Agrícola de la Universidad Nacional de Colombia en Bogotá, Colombia, tuvo como objetivo principal desarrollar un programa de computador para el análisis lineal de estructuras aporticadas tridimensionales bajo cargas estáticas. La investigación involucró el uso del método de rigideces y la aplicación de cuaterniones para el cálculo de la matriz de transformación de rotación, utilizando las tecnologías web HTML, CSS, y JavaScript. El programa desarrollado, mas.js, se complementó con una librería de análisis estructural llamada pymas, ambas capaces de interactuar mediante archivos en formato JSON para intercambiar información estructural y análisis. Los resultados principales indicaron que el software desarrollado proporcionaba una representación visual eficiente y un análisis preciso de estructuras tridimensionales, permitiendo a los usuarios realizar análisis desde cualquier navegador web sin necesidad de instalar software adicional. En conclusión, el software es una

herramienta innovadora para el análisis estructural, demostrando que las tecnologías web pueden ofrecer soluciones accesibles y potentes en el campo de la ingeniería estructural. La relevancia de esta tesis para la presente investigación radica en el enfoque hacia el desarrollo de software estructural accesible y modular, lo cual respalda la filosofía de OpenRSE en cuanto a accesibilidad y eficiencia en el análisis estructural bidimensional.

Asmal Rodas y Yumbla Cadme (2023), en su tesis titulada "Implementación y validación de un programa basado en el método de elementos finitos para la solución de problemas cuasi-estáticos en cuerpos bidimensionales", presentada en la Facultad de Ingeniería de la Universidad de Cuenca en Cuenca, Ecuador, tuvo como objetivo implementar y validar un programa basado en el método de elementos finitos (MEF) para resolver problemas cuasi-estáticos en cuerpos bidimensionales. La investigación involucró la modificación y adaptación de un software de código abierto llamado FSIPY, que fue depurado y programado para mejorar su precisión y aplicabilidad. Se utilizaron herramientas técnicas como GMSH para el mallado y ParaView para la visualización de los resultados. Los resultados principales indicaron que el software modificado alcanzó una alta precisión, con una diferencia relativa del 1% en términos de desplazamientos comparado con los resultados obtenidos en software comerciales como SAP2000 y ABAQUS, demostrando así su confiabilidad. En conclusión, la tesis valida el uso de FSIPY como una herramienta efectiva para resolver problemas cuasi-estáticos en cuerpos bidimensionales, validando su precisión mediante comparaciones con SAP2000, un enfoque que también será utilizado en la presente investigación para validar OpenRSE. La relevancia de esta tesis para la investigación actual radica en la metodología empleada para validar un software estructural mediante la comparación con SAP2000, lo cual coincide con el enfoque que se utilizará para comparar y validar OpenRSE junto con SAP2000 y Robot Structural. Esto refuerza la importancia de los procedimientos de validación al crear herramientas de análisis estructural.

2.1.2. ANTECEDENTES NACIONALES

Díaz Barrantes y Guillén Hernández (2020), en su tesis titulada "Modelo computacional para el análisis matricial de estructuras reticulares", presentada en la Facultad de Ingeniería de la Universidad Peruana de Ciencias Aplicadas en Lima, Perú, tuvo como objetivo desarrollar un modelo computacional basado en el método matricial para el análisis estructural de estructuras reticulares, tales como armaduras y pórticos en dos dimensiones. La investigación involucró la implementación del programa en el entorno de desarrollo Matlab, utilizando su lenguaje de programación propio para crear una herramienta educativa dirigida principalmente a estudiantes de ingeniería civil. Este modelo se diseñó para facilitar la comprensión y ejecución de cálculos estructurales en armaduras y pórticos planos, sin depender de software comercial, y con el objetivo de que futuros desarrollos lo expandieran a un software más avanzado. La investigación se basó en metodologías computacionales utilizando el método matricial de rigidez para el análisis estructural. El software desarrollado permitía visualizar desplazamientos nodales, fuerzas internas y reacciones en los elementos estructurales. Se realizaron pruebas de validación comparando los resultados con los obtenidos mediante el software comercial de Aslam Kassimali, mostrando un margen de error inferior al 0.05%, lo que demostró la precisión del modelo. Los resultados indicaron que el software desarrollado es capaz de realizar análisis precisos en estructuras reticulares, proporcionando visualizaciones gráficas que facilitaban la interpretación de los resultados.En conclusión, la tesis creó una herramienta útil y eficiente para el análisis de estructuras reticulares, demostrando que es posible desarrollar software de alta precisión en un entorno académico sin depender de software comercial. La relevancia de esta tesis para la presente investigación radica en su enfoque en el desarrollo de un programa basado en el método de análisis matricial, que en este caso se centra en estructuras reticulares. Esto es particularmente relevante, ya que en la presente investigación se busca desarrollar OpenRSE, un

software que también utilizará el análisis matricial, pero enfocado en una mayor diversidad de modelos estructurales, tanto rígidos como reticulares, abarcando hasta estructuras hiperestáticas. Este antecedente respalda el objetivo de crear un software que sea accesible y eficiente, validado frente a herramientas comerciales, lo que fortalece la idea de que un programa académico puede cumplir con los altos estándares necesarios en el análisis estructural.

Lupaca Quispe (2022), en su tesis titulada "Desarrollo del simulador web Beli para el análisis matricial de estructuras planas y espaciales", presentada en la Facultad de Ingeniería de la Universidad César Vallejo en Lima, Perú, tuvo como objetivo optimizar el análisis matricial de estructuras planas y espaciales mediante el desarrollo de un simulador web llamado Beli. Este simulador está diseñado para ser accesible desde cualquier dispositivo con conexión a internet, utilizando tecnologías como HTML, CSS y JavaScript para su implementación. La investigación involucró principalmente el análisis de estructuras como armaduras, vigas continuas y pórticos, validando la precisión del simulador al compararlo con SAP2000, uno de los softwares comerciales más utilizados en el análisis estructural. La metodología utilizada fue de tipo aplicada, con un diseño pre-experimental, lo que implicó la validación de la herramienta desarrollada mediante la comparación de sus resultados con los obtenidos por SAP2000. El simulador permitió a los usuarios ingresar datos estructurales, como las propiedades de materiales, geometría y cargas, para luego generar los resultados de desplazamientos, reacciones y fuerzas internas. Los resultados principales indicaron que las diferencias en los cálculos realizados por Beli en comparación con SAP2000 eran mínimas. Por ejemplo, para armaduras, la variación máxima en los desplazamientos fue de solo 1E-06 metros, lo que demuestra la precisión del simulador. En conclusión, el simulador Beli optimizó el análisis estructural de estructuras planas y espaciales, brindando una alternativa accesible y precisa frente a los costosos programas comerciales. Los resultados obtenidos fueron satisfactorios y comparables con los de SAP2000, lo que valida la

fiabilidad del simulador. La relevancia de esta tesis para la presente investigación radica en su enfoque en el desarrollo de un programa computacional basado en el método matricial. Al igual que OpenRSE, Beli busca ofrecer una herramienta accesible para el análisis estructural, pero con la ventaja adicional de su implementación web. Este trabajo también destaca la importancia de validar los resultados frente a software comerciales, lo que es fundamental para el desarrollo y la validación de OpenRSE, cuyo objetivo es analizar distintos tipos de estructuras, incluidas las hiperestáticas, mejorando la accesibilidad y eficiencia en la ingeniería estructural.

Montúfar Chata (2022), en su tesis titulada "Análisis comparativo del modelamiento y diseño estructural en concreto armado utilizando los softwares SAP2000, ETABS, CYPECAD y Revit Structure, para la infraestructura educativa Sorapa", presentada en la Facultad de Ingeniería Civil y Arquitectura de la Universidad Nacional del Altiplano en Puno, Perú, tuvo como objetivo comparar los resultados del modelamiento y diseño estructural en concreto armado de la infraestructura educativa Sorapa utilizando diferentes programas de software. Los programas evaluados incluyeron SAP2000, ETABS, CYPECAD y Revit Structure. El propósito de esta investigación fue determinar cuál de estos programas proporcionaba mejores resultados en términos de optimización de diseño, eficiencia de cálculo y ajuste a las normativas peruanas de construcción. La metodología involucró el modelamiento estructural de la infraestructura educativa, comenzando con la elaboración de planos arquitectónicos, predimensionamiento de los elementos estructurales y la asignación de cargas y combinaciones de diseño según los criterios establecidos por el Reglamento Nacional de Edificaciones (RNE). Posteriormente, se realizaron los cálculos estructurales en cada uno de los softwares, evaluando el comportamiento dinámico y estático de la edificación bajo cargas sísmicas, de acuerdo con la Norma E.030. Los resultados de cada software fueron comparados en términos de desplazamientos, fuerzas internas, diseño de acero y costos estimados. Los resultados principales

indicaron que ETABS fue el software que proporcionó un diseño más eficiente en cuanto al cálculo de acero, con una reducción del 4% en comparación con SAP2000, y variaciones menores frente a CYPECAD y Revit Structure. Estas diferencias se atribuyeron principalmente a las metodologías de cálculo de cada software y su enfoque particular para el análisis de edificios educativos. Adicionalmente, se observó que Revit Structure facilitaba la integración con otras disciplinas de diseño, como la arquitectura y las instalaciones, mientras que SAP2000 y ETABS ofrecían mejores resultados en el análisis estructural puro. En conclusión, el estudio demostró que el uso de diferentes programas de software puede arrojar variaciones significativas en el diseño estructural y los costos asociados. La relevancia de esta tesis para la presente investigación radica en su enfoque en la comparación de diferentes herramientas de software para el análisis estructural, todas las cuales son programas privativos y costosos que representan las herramientas típicas utilizadas en la industria. Esto resalta la necesidad de alternativas accesibles, como OpenRSE, que no solo busca ser una solución libre y gratuita, sino también ofrecer la transparencia de un software de código abierto, donde el método de cálculo puede ser verificado y adaptado a las necesidades específicas de los usuarios. A pesar de que los cuatro programas utilizaron el mismo modelo estructural, los resultados presentaron ligeras variaciones, lo que refuerza la importancia de poder personalizar y verificar el software para garantizar que el cálculo se ajuste a las exigencias del proyecto y a las preferencias del usuario.

2.1.3. ANTECEDENTES LOCALES

Inza Ramírez y Nación Ramos (2024), en su tesis titulada "Diseño estructural de la superestructura y subestructura de un puente tipo viga-losa ubicado en el centro poblado de Jancao Bajo", presentada en la Facultad de Ingeniería Civil y Arquitectura de la Universidad Nacional Hermilio Valdizán en Huánuco, Perú, tuvo como objetivo diseñar la superestructura y subestructura de un puente tipo viga-losa de 12 metros de luz. La investigación involucró un análisis detallado de las cargas

aplicadas al puente, siguiendo las normas del Manual de Puentes del Ministerio de Transportes y Comunicaciones (MTC) de Perú y utilizando el software SAP2000 para la verificación de los cálculos manuales. La metodología se centró en tres fases: predimensionamiento de la losa y las vigas, análisis de las cargas, y diseño estructural de los componentes principales del puente, incluyendo la losa, vigas, y la berma. Se realizó un meticuloso análisis comparativo entre los cálculos manuales y los resultados obtenidos con SAP2000, lo que permitió validar la precisión del diseño. Los resultados principales indicaron que las dimensiones finales de los elementos estructurales fueron satisfactorias, con una variación mínima en los momentos flectores entre los cálculos manuales y los obtenidos por el software, siendo la mayor discrepancia de apenas 0.5%. En cuanto a los materiales, se especificó el uso de acero reforzado en las zonas más críticas del puente. En conclusión, la investigación demostró que el diseño estructural propuesto cumple con los estándares técnicos del MTC y que el uso del software SAP2000 permite una verificación confiable de los cálculos manuales, asegurando la precisión y seguridad del puente. El trabajo proporcionó un diseño completo y detallado de un puente que mejora la infraestructura vial y la seguridad en la región. La relevancia de esta tesis para la presente investigación radica en su enfoque en el diseño estructural basado en el método de elementos finitos, validado mediante el uso de software especializado. Además, como en muchas otras investigaciones sobre análisis de estructuras, los autores recurren al uso de SAP2000 o ETABS para la verificación, ya que son los programas más comunes cuando se trata de puentes o edificaciones. Esto pone en evidencia las pocas alternativas disponibles en el mercado, dado que estos programas, si bien son altamente confiables, son de alto costo, lo que limita su accesibilidad para muchos usuarios y proyectos. Por lo tanto, este antecedente refuerza la importancia de desarrollar alternativas libres y gratuitas como OpenRSE, que pueda ofrecer una solución de bajo costo, pero con la misma precisión y eficiencia, brindando más accesibilidad a ingenieros y estudiantes.

Lopez Pajuelo (2022), en su tesis titulada "Propuesta de diseño estructural de un reservorio aplicando el software SAP2000 para el servicio de agua potable del centro poblado de Matibamba, Amarilis-2022", presentada en la Facultad de Ingeniería Civil de la Universidad de Huánuco en Huánuco, Perú, tuvo como objetivo realizar el diseño estructural de un reservorio de agua potable utilizando el software SAP2000, con el fin de asegurar el suministro adecuado y resistente a sismos para el centro poblado de Matibamba. La investigación se centró en analizar el comportamiento sísmico del reservorio y diseñarlo de acuerdo con las normativas peruanas y el código internacional ACI 350. La metodología consistió en el predimensionamiento de los componentes estructurales del reservorio, como la cúpula, la viga anular, el muro y la losa de fondo, seguidos del modelamiento de la estructura en SAP2000, donde se incorporaron factores sísmicos, tales como la masa convectiva e impulsiva del agua almacenada. Los cálculos consideraron el peso de los elementos estructurales, así como la presión ejercida por el agua. Los resultados obtenidos indicaron que las dimensiones y el refuerzo estructural diseñados cumplen con las normativas aplicables, y los análisis dinámicos y estáticos demostraron que el reservorio puede soportar las fuerzas sísmicas previstas. En conclusión, el diseño estructural propuesto cumple con los requisitos del Reglamento Nacional de Edificaciones (RNE) y las normativas ACI 350 para estructuras de almacenamiento de agua. Se concluye que el uso del software SAP2000 permitió un diseño preciso, optimizando la cantidad de acero necesaria para garantizar la seguridad y durabilidad del reservorio frente a cargas dinámicas y estáticas. La relevancia de esta tesis para la presente investigación radica en su enfoque en el uso de SAP2000 para el diseño estructural y su aplicación en el análisis de cargas sísmicas en reservorios, lo cual es esencial para la validación de modelos estructurales. Además, este trabajo destaca la necesidad de herramientas accesibles y eficientes para el análisis estructural en proyectos de infraestructura crítica, similar a lo que OpenRSE busca lograr, ofreciendo una alternativa más accesible frente a los altos costos

de licencias de software comercial como SAP2000, demostrando la importancia de desarrollar herramientas de código abierto.

Guerra Utrilla (2022), en su tesis titulada "Análisis y diseño estructural con el software ETABS de un edificio comercial de 5 niveles de concreto armado, Huánuco - Huánuco -2022", presentada en la Facultad de Ingeniería Civil de la Universidad de Huánuco en Huánuco, Perú, tuvo como objetivo realizar el análisis y diseño estructural de un edificio comercial de cinco niveles, empleando el software ETABS para asegurar la seguridad y eficiencia del diseño estructural conforme a las normas locales e internacionales de construcción, tales como el Reglamento Nacional de Edificaciones (RNE) y el código ACI 318-14. La investigación involucró el análisis de cargas vivas y muertas, así como la implementación de cargas sísmicas de acuerdo con la Norma E030, lo que permitió estudiar el comportamiento dinámico y estático de la estructura. Para ello, se utilizó ETABS, un software especializado en modelado y análisis estructural, que permitió la simulación y diseño de todos los componentes estructurales del edificio, como vigas, columnas, losas y muros de corte. La metodología incluyó el modelamiento de la estructura, el análisis de los resultados obtenidos mediante ETABS y la verificación de los elementos estructurales con base en la normativa aplicable. Los resultados principales indicaron que el uso de muros de corte redujo significativamente las derivas elásticas y mejoró la rigidez del edificio, disminuyendo el período de vibración y asegurando que la estructura sea capaz de resistir fuerzas sísmicas dentro de los límites establecidos por las normativas. En particular, el análisis dinámico reveló un periodo de vibración de 0.314 segundos, lo que confirma la estabilidad de la edificación bajo condiciones de sismo. En conclusión, el trabajo demostró que la aplicación del software ETABS es efectiva para diseñar edificios comerciales que cumplan con los estándares de seguridad estructural, proporcionando una herramienta que optimiza tanto el tiempo como los recursos invertidos en el diseño. La relevancia de esta tesis para la presente investigación radica en el uso de ETABS para modelar y validar el comportamiento estructural de edificaciones

bajo cargas sísmicas. De manera similar, en el desarrollo de OpenRSE, se busca proporcionar una herramienta que permita realizar análisis estructurales con un enfoque en la accesibilidad y la eficiencia, ofreciendo una alternativa frente a los altos costos de software comerciales como ETABS y SAP2000, especialmente en contextos donde se necesita una herramienta de código abierto.

2.2. BASES TEÓRICAS

2.2.1. VARIABLE DEPENDIENTE

Eficiencia en el Software de Análisis Estructural

La eficiencia en el análisis estructural puede definirse por la simplicidad y rapidez con la que se plantea un modelo estructural, la efectividad de su análisis y el control adecuado de los errores. En este contexto, un sistema eficiente no solo requiere un método de análisis confiable, sino también un proceso bien estructurado desde la entrada de datos hasta la interpretación de los resultados. Según Celigueta (1998), el análisis estructural busca determinar el estado de deformaciones y tensiones que se producen en una estructura debido a las acciones que actúan sobre ella. Para obtener resultados efectivos, este proceso requiere una idealización precisa de la estructura, donde se determinan las dimensiones de los elementos y las fuerzas que actúan sobre ellos. Sin embargo, la eficiencia del análisis no solo se relaciona con la precisión, sino también con la rapidez en la ejecución del análisis y su capacidad de manejar diversas condiciones de carga (Celigueta, 1998, pp. 1-4).

La integración de nuevas tecnologías, como el modelado 3D y la simulación por elementos finitos, ha permitido optimizar los diseños estructurales. Herramientas avanzadas como Tekla Structural Designer y SAP2000 se han convertido en estándares para mejorar la eficiencia del proceso de diseño, permitiendo a los ingenieros analizar una

estructura en diferentes escenarios antes de su construcción (Construsoft, 2024). Estas herramientas automatizan la generación de modelos analíticos y optimizan el uso de materiales, lo que ahorra tiempo y mejora la precisión del diseño.

Además, el uso de herramientas como Tekla no solo optimiza el análisis de estructuras de acero y hormigón, sino que también permite generar documentación detallada y actualizada automáticamente conforme el modelo evoluciona (Construsoft, 2024). Esto minimiza errores y mejora la coherencia en el proceso de diseño, fortaleciendo la relación entre la eficiencia y el control de calidad en proyectos complejos.

Por su parte, Luthe (s.f.) destaca el impacto que ha tenido el desarrollo de la tecnología informática en la eficiencia del análisis estructural. El autor resalta cómo el uso de métodos matriciales, facilitado por las computadoras, ha permitido simplificar y agilizar el proceso de análisis, haciendo posible el estudio de estructuras complejas con múltiples sistemas de carga. Aunque estos métodos tienen un origen en los planteamientos manuales tradicionales, la eficiencia de los métodos computacionales radica en la rapidez con la que se pueden realizar cálculos complejos y su capacidad para generar resultados precisos en poco tiempo (Luthe, s.f., pp. ix-x).

En mi investigación, me ceñiré principalmente a la definición de eficiencia propuesta por Luthe, debido a que su enfoque en los métodos matriciales es fundamental para el desarrollo de OpenRSE. Estos métodos permiten abordar el análisis estructural bidimensional con rapidez y simplicidad, lo que es coherente con los objetivos de mi trabajo. Sin embargo, la importancia de la idealización precisa de las estructuras, como lo indica Celigueta (1998), también será clave en la implementación del programa, asegurando un control adecuado de errores y resultados confiables.

Accesibilidad en el Software de Análisis Estructural

La accesibilidad en el contexto del software de ingeniería se refiere a la facilidad con la que los usuarios pueden acceder, entender y utilizar las herramientas disponibles, sin que las barreras técnicas o económicas limiten su uso. En el campo del análisis estructural, la accesibilidad adquiere una importancia crítica debido a la necesidad de que tanto profesionales como estudiantes puedan contar con programas que faciliten el diseño y cálculo de estructuras, ya que estos procesos son esenciales en la construcción y la ingeniería civil.

Una de las barreras más comunes en el acceso al software especializado es el alto costo. Programas como CYPECAD, SAP2000 o Autodesk Robot requieren inversiones significativas. Por ejemplo, CYPECAD está incluido en el flujo de trabajo Open BIM. Su precio oscila de 1.100 a 9.998 €, desde el más básico al más completo (Structuralia, 2021). Este tipo de costos puede ser prohibitivo para muchos profesionales y estudiantes, especialmente en contextos donde los salarios no son proporcionales a los costos de este tipo de software. En este sentido, la relación entre los costos del software y los ingresos en ciertos contextos geográficos hace que la accesibilidad sea limitada.

Además del costo, existen barreras técnicas, como la complejidad de algunos programas y los requisitos de hardware necesarios para ejecutarlos. Por ejemplo, SAP2000 es un software ampliamente utilizado para dimensionar puentes, presas, edificios y todo tipo de infraestructuras y cuenta con una interfaz gráfica 3D orientada a objetos, pero su uso requiere computadoras con capacidades avanzadas, lo cual puede ser otra barrera para aquellos que no disponen de recursos tecnológicos adecuados (Incober, 2024).

En mi investigación, el desarrollo de OpenRSE busca precisamente mejorar la accesibilidad al análisis estructural bidimensional, proporcionando una herramienta más accesible tanto en términos de

costo como de simplicidad de uso. Esta iniciativa pretende derribar las barreras de acceso que limitan a muchos usuarios, ofreciendo un software libre y portable que pueda ser utilizado en una amplia gama de dispositivos y sin los altos costos de las licencias tradicionales. De esta manera, se busca fomentar la igualdad de oportunidades para estudiantes y profesionales, facilitando su acceso a herramientas avanzadas sin depender de altos recursos económicos o técnicos.

Interrelación entre Eficiencia y Accesibilidad

La eficiencia y la accesibilidad en el software de análisis estructural están intrínsecamente relacionadas. La eficiencia, en términos de tiempo y precisión, se ve influenciada directamente por las herramientas disponibles y cómo se accede a ellas. Cuando un software es accesible, no solo en términos de costo, sino también en su facilidad de uso y disponibilidad, los procesos de diseño y análisis estructural se ven optimizados.

Como menciona González Cárceles (1990), los apoyos son cada vez más poderosos debido a la progresiva utilización de las máquinas, que permiten la manipulación de cantidades crecientes de variables de forma simultánea (p. 1). Esto significa que el uso de herramientas avanzadas facilita la resolución de problemas complejos, mejorando significativamente la eficiencia del proceso. Sin embargo, es crucial que estas herramientas sean accesibles para que un mayor número de usuarios pueda beneficiarse de dichas ventajas. La accesibilidad, en este sentido, no se limita únicamente al costo, sino que abarca la facilidad de aprender y utilizar estas herramientas, lo que a su vez potencia la eficiencia.

La relación entre accesibilidad y eficiencia se refuerza cuando se consideran las barreras que enfrentan tanto profesionales como estudiantes al utilizar software de análisis estructural. Programas costosos y técnicamente complejos pueden limitar el acceso a

herramientas eficientes, lo que contrasta con el objetivo de diseñar las estructuras más que analizarlas (González Cárceles, 1990, p. 2). En este sentido, cuando un programa es accesible y está diseñado de forma intuitiva, permite a los usuarios centrarse en la calidad del diseño, minimizando los errores que pueden surgir en el proceso.

En mi investigación, OpenRSE tiene como objetivo abordar estos desafíos, mejorando tanto la accesibilidad como la eficiencia del análisis estructural bidimensional. La accesibilidad de OpenRSE no solo se manifestará en su gratuidad, sino también en su diseño intuitivo y en la posibilidad de ser utilizado en una amplia gama de dispositivos, permitiendo que los usuarios se concentren en la precisión de los cálculos, como sugiere González Cárceles (1990), quien señala que un diseño inteligente precisa analizar el propio procedimiento para minimizar cambios innecesarios en el análisis y diseño (p. 7).

Al facilitar el acceso a herramientas que permiten un diseño y análisis más rápido y preciso, se logra un proceso más eficiente. La eficiencia, en este caso, no se trata solo de resolver cálculos complejos, sino de minimizar el riesgo de los cambios a que obliga el análisis sobre valores ya decididos (González Cárceles, 1990, p. 7). En este sentido, OpenRSE busca integrar estos principios para mejorar el diseño estructural y hacer que sea accesible a un mayor número de profesionales y estudiantes.

2.2.2. VARIABLE INDEPENDIENTE

¿Qué es el método matricial?

El método matricial es una herramienta fundamental en el análisis estructural moderno, especialmente debido a su capacidad para manejar sistemas de ecuaciones complejas con múltiples variables, sin perder información significativa durante el proceso. Blanco Claraco, González Herrera y García-Manrique Ocaña (2012) definen el cálculo matricial

(CM) como un conjunto de métodos que organizan toda la información en forma de matrices y resaltan que las relaciones entre las distintas partes de una estructura se establecen sin suposiciones que eliminen información relevante (p. 9). Este enfoque es clave para garantizar que los cálculos estructurales sean precisos y se puedan resolver de manera automática mediante el uso de software especializado, lo que ha hecho que este método sea de uso común en la ingeniería moderna.

Uno de los aspectos más relevantes del método matricial es su generalidad y aplicabilidad a diferentes tipos de estructuras. Los métodos clásicos, como los usados en estructuras articuladas o el método de Hardy Cross, requieren simplificaciones que pueden reducir la precisión de los resultados. En contraste, el CM se puede considerar un método de cálculo general y no está limitado a una tipología estructural particular, lo que lo convierte en una herramienta versátil para los ingenieros (Blanco Claraco et al., 2012, p. 10). Además, los autores destacan que una de las principales ventajas del método es su capacidad para incluir todas las ecuaciones del sistema, evitando el descarte de variables que, aunque aparentemente insignificantes, pueden influir en los resultados finales.

En términos de velocidad y automatización, el método matricial ha sido revolucionario. Aunque es cierto que el tiempo de cálculo es considerablemente mayor que en los métodos tradicionales, el uso de computadoras ha permitido que este problema se solvente, logrando cálculos rápidos y precisos (Blanco Claraco et al., 2012). Esta automatización ha sido especialmente útil con el método de la rigidez, el cual organiza la información de tal manera que permite desarrollar algoritmos de aplicación automática para una gran variedad de estructuras, como subrayan los autores al afirmar que el método de la rigidez [...] ha permitido la implantación de programas de ordenador que aprovechan al máximo las ventajas del CM (Blanco Claraco et al., 2012, p. 12).

En cuanto a la discretización, un aspecto clave del análisis estructural moderno, Celigueta Lizarza (2008) destaca que los sistemas discretos permiten definir la deformación de una estructura mediante un número finito de parámetros, lo que simplifica su análisis (p. 1). Esta aproximación es esencial para el método de los elementos finitos, que amplía las capacidades del cálculo matricial al permitir modelar estructuras complejas mediante la división en pequeños elementos finitos. Como consecuencia, se pueden analizar sistemas continuos, como losas de cimentación o estructuras complejas, que serían difíciles de resolver utilizando métodos analíticos tradicionales (Celigueta Lizarza, 2008, p. 2).

El método matricial ha transformado el análisis estructural al permitir la automatización de cálculos complejos y al brindar un enfoque general y preciso para diferentes tipos de estructuras. Su relevancia en la ingeniería moderna se debe, en gran medida, a la capacidad de manejar grandes sistemas de ecuaciones con la ayuda de herramientas computacionales, permitiendo a los ingenieros centrarse en la modelización y el diseño, mientras el software se encarga de los cálculos numéricos.

Método matricial de rigidez estructural

El método matricial de rigidez estructural tiene como base la formulación matemática que relaciona las fuerzas aplicadas a una estructura con los desplazamientos en los nodos, a través de la matriz de rigidez. Este enfoque es fundamental en el análisis estructural moderno, ya que permite modelar de manera precisa el comportamiento de estructuras sometidas a diversas cargas.

Quispe Panca (2015) señala que realizar el procedimiento de cálculo para determinar los esfuerzos internos y externos implica cuantificar en sus respectivas unidades lo siguiente: ε (deformaciones unitarias del medio continuo), σ (esfuerzos asociados a las

deformaciones sufridas por el medio continuo) y δ (desplazamientos de los diferentes puntos del medio continuo que forma la estructura) (p. 18). Estas unidades básicas, al relacionarse entre sí, conducen a la ecuación fundamental del método matricial de rigidez:

$$\{F\} = [\ K\ ] * \{\delta\} \tag{1}$$

donde $\{F\}$ es el vector de fuerzas externas, $[K]$ es la matriz de rigidez del sistema y $\{\delta\}$ es el vector de desplazamientos de los nudos.

Por otro lado, Blanco Claraco et al. (2012) explican que la ecuación fundamental también se puede derivar relacionando los principios de equilibrio, comportamiento y compatibilidad, lo que lleva a una forma similar:

$$F = K * U \tag{2}$$

(p. 14). Aunque ambos autores describen el mismo concepto, se observa una diferencia en la notación del vector de desplazamientos. En esta investigación, se utilizará la notación U para denotar los desplazamientos nodales, siguiendo la convención propuesta por Blanco Claraco et al. (2012).

Por lo tanto, la ecuación fundamental del método matricial de rigidez estructural se define de la siguiente manera:

$$\{F\} = [\ K\ ] * \{U\} \tag{3}$$

Donde:

$\{F\}$: Vector de fuerzas externas

$[K]$: Matriz de rigidez del sistema

$\{U\}$: Vector de desplazamiento en los nudos

Sistema de referencia

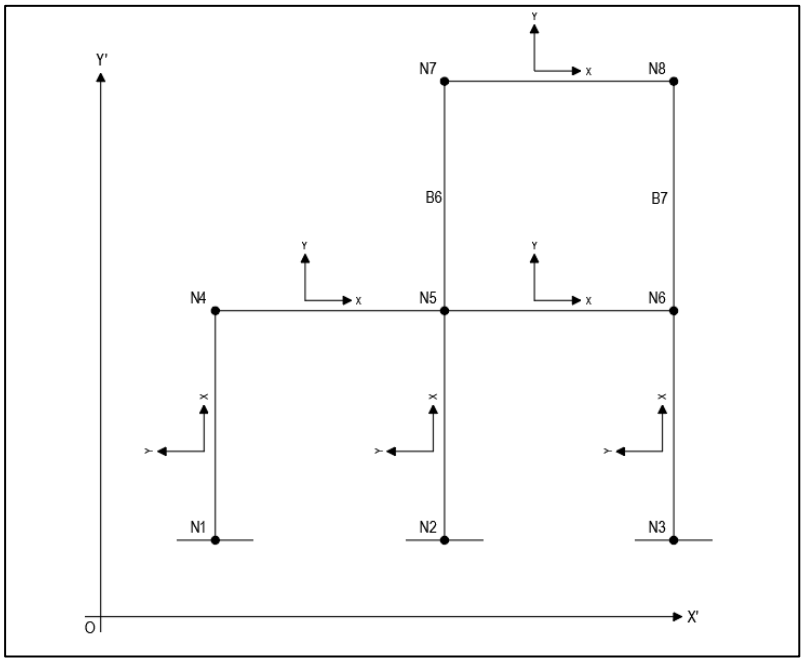
Rojas Rojas y Padilla Punzo (2009) y Quispe Panca (2015) coinciden en la importancia de establecer dos sistemas de referencia en el análisis estructural. El primero es el sistema de coordenadas locales (ejes x, y, z). Este sistema se refiere a las propiedades del elemento, como las dimensiones y los momentos de inercia. La dirección del sistema de coordenadas locales queda determinada por la configuración del elemento estructural, que comienza en un nodo de origen y termina en un nodo final, formando así una barra cuya dirección principal se sitúa a lo largo del eje x, el cual sigue el eje centroidal del elemento.

El segundo sistema de referencia es el sistema de coordenadas globales (ejes x', y', z'). Este sistema se aplica a todos los elementos de la estructura en su conjunto, incluyendo los nodos y las cargas que actúan sobre ellos. Las coordenadas globales permiten unificar el análisis de los distintos elementos de la estructura en un mismo marco de referencia, facilitando así el proceso de ensamblaje de matrices y la solución del problema estructural.

Este enfoque en la distinción entre coordenadas locales y globales es fundamental para garantizar que las propiedades de cada elemento se calculen correctamente en su sistema propio (local), mientras que las interacciones entre elementos se evalúan en el sistema de coordenadas globales (Rojas Rojas & Padilla Punzo, 2009).

Figura 1

Sistema de coordenadas locales y globales



Nota. Adaptado de Análisis matricial de estructuras: Introducción al método de elementos finitos (Figura 2.2, p. 6), por A. J. Quispe Panca, 2015, Macro EIRL.

Es crucial entender que, para que un sistema de ecuaciones funcione correctamente, todo debe operar en un solo sistema de coordenadas. De lo contrario, los resultados no tendrían sentido o serían extremadamente complicados de interpretar. Por esta razón, es necesario realizar una transformación de coordenadas. Pensemos en lo siguiente: una estructura está formada por múltiples barras, y como se mencionó anteriormente, cada barra o elemento tiene su propio sistema de coordenadas local. Para trabajar con estos elementos de manera coherente, es necesario transformar sus sistemas de coordenadas locales a las coordenadas globales. Esto permite relacionar adecuadamente la posición de los nodos y las cargas asignadas, lo cual es fundamental para iniciar el análisis estructural.

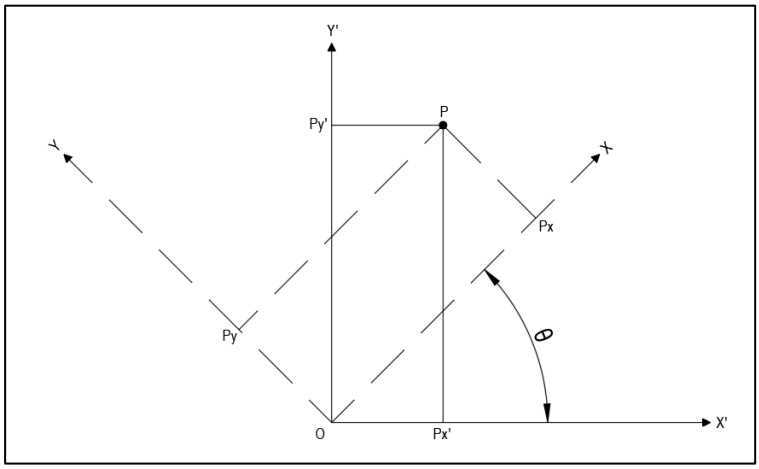
Blanco Claraco et al. (2012) indican que la relación entre ambos pares de coordenadas se puede establecer fácilmente mediante relaciones trigonométricas, conociendo únicamente el giro o ángulo de

inclinación (positivo en la dirección contraria a las agujas del reloj). De manera similar, Quispe Panca (2015) simplifica esta idea señalando que, para cualquier barra, se utiliza el ángulo theta para denotar la inclinación del eje local respecto al eje horizontal global.

Al examinar estas afirmaciones, observamos que ambos autores usan simbologías diferentes para el ángulo de inclinación de una barra. En esta investigación, seguiremos la convención de Quispe Panca y utilizaremos θ para el ángulo. Además, usaremos la convención de Rojas Rojas y Padilla Punzo para las coordenadas locales como (x, y) y las globales como (x', y'), por preferencia personal, como se muestra en la Figura 2.

Figura 2

Sistema de referencia



Nota. Adaptado de Análisis estático de estructuras por el método matricial (Figura 1.7.1, p. 23), por J. L. Blanco Claraco, A. González Herrera, & J. M. García-Manrique Ocaña, 2012, Universidad Málaga.

Supongamos un punto en el espacio P. En un sistema local bidimensional, este punto se denota por el par ordenado (x, y). Como se mencionó en párrafos anteriores, es necesario establecer este punto en un sistema de coordenadas globales (x', y'). Rojas Rojas y Padilla Punzo (2009) y Quispe Panca (2015) indican que la conversión de coordenadas

locales a globales se puede realizar fácilmente mediante el uso de relaciones trigonométricas, siempre que se conozca el ángulo de rotación y las coordenadas locales del punto en cuestión. Este proceso es fundamental para el análisis estructural, ya que permite integrar y relacionar adecuadamente todos los elementos de una estructura dentro de un sistema de coordenadas global, como se explicó anteriormente.

Al conocer el ángulo de rotación (θ), es posible determinar las nuevas coordenadas de cualquier punto en el sistema global. Este método facilita la transformación y el análisis de múltiples elementos que conforman una estructura compleja. Las ecuaciones proporcionadas por ambos autores para esta conversión son las siguientes:

$$x' = x * \cos(\theta) - y * sin(\theta) \quad (4)$$

$$y' = x * \sin(\theta) + y * cos(\theta) \quad (5)$$

Lo que matemáticamente no es más que una combinación lineal de las coordenadas locales, lo que permite representar estas ecuaciones en forma matricial (Blanco Claraco et al., 2012), obteniendo lo siguiente:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad (6)$$

La ecuación 4 describe la transformación de un punto P de coordenadas locales a globales, utilizando un par ordenado. Además de hablar directamente en términos de pares ordenados, también estamos usando notación vectorial, lo que representa mejor un vector. La ecuación simplificada queda de la siguiente manera:

$$\{x', y'\} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \{x, y\} \quad (7)$$

Por último, de esta ecuación se deduce que al multiplicar un par ordenado por la matriz indicada en la ecuación 5, transformamos las

coordenadas de locales a globales. A esta matriz de transformación la denominaremos $[T]$. Finalmente, se establece lo siguiente:

$$[T] = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \quad (8)$$

En el caso de que no sea suficiente trabajar únicamente con un vector de desplazamiento en (x, y), sino que también debamos considerar la rotación (θ), es importante entender que este ángulo corresponde a la flexión que experimenta una barra en uno de sus extremos. En términos simples, este ángulo de rotación o flexión tiene el mismo valor tanto en el sistema local como en el global, ya que no está determinado por el par ordenado (x, y). Por lo tanto, esta identidad se representa de la siguiente manera (Blanco Claraco et al., 2012):

$$[T] = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (9)$$

La ecuación 7 transforma un par ordenado de un sistema local a global. Adicionalmente, también será necesario llevar un sistema global a local. Quispe Panca (2015) indica lo siguiente: en esta matriz (referente a la ecuación 7), se tiene una característica importante que consiste en que el determinante de la matriz es igual a $\cos\text{^}2(\theta) + \sin\text{^}2(\theta) = 1$, lo cual implica que es una matriz ortogonal donde se cumple:

$$[T^{-1}] = [T^T] \quad (10)$$

De esta manera, establecemos la matriz de transformación de global a local. Al evaluar la expresión de la ecuación 8, obtenemos el siguiente resultado:

$$[T^T] = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (11)$$

para integrar este elemento en el análisis estructural global, es necesario transformar su matriz de rigidez local k al sistema de coordenadas global. Esto se realiza utilizando una matriz de transformación T, que convierte las propiedades del elemento desde el sistema local al sistema global. El cálculo de la matriz de rigidez global del elemento K se lleva a cabo mediante el siguiente triple producto matricial (Aguiar Falconi, 2004, p. 395):

$$K = T^T * k * T \tag{12}$$

Este procedimiento garantiza que la matriz de rigidez local k del elemento se transforme adecuadamente al sistema global. La matriz global K del elemento permite relacionar correctamente las fuerzas globales F y los desplazamientos globales U de todo el sistema estructural. De esta manera, se asegura la coherencia en el análisis estructural mediante la siguiente ecuación fundamental

La aplicación de esta transformación es crucial para ensamblar la matriz de rigidez global de la estructura, ya que cada elemento, inicialmente definido en sus propias coordenadas locales, debe expresarse en el sistema global para que el análisis estructural sea consistente. A través de esta transformación, no solo se integran las matrices de rigidez, sino también los vectores de desplazamientos y fuerzas, permitiendo un análisis global preciso.

Grados de libertad

Según Quispe Panca (2015), Los grados de libertad son los desplazamientos independientes (traslaciones U y rotaciones R) de los nodos que son necesarios para especificar la forma deformada de una estructura cuando se somete a una carga arbitraria (p. 25). Esta descripción define claramente el concepto de grado de libertad, referido a los movimientos que un nodo puede experimentar dentro de una estructura.

Es importante destacar que un nodo es libre cuando puede desplazarse o rotar en una dirección específica; de lo contrario, se dice que está restringido. En un análisis estructural, los GDL juegan un papel clave, ya que determinan cómo se comporta la estructura frente a cargas aplicadas.

Consideremos un nodo n_i en un espacio bidimensional. Este nodo puede tener tres posibles movimientos: traslación en el eje x, traslación en el eje y y rotación alrededor de su propio eje. Por lo tanto, se dice que un nodo en un espacio bidimensional tiene tres grados de libertad (GDL): dos traslaciones y una rotación. Estos GDL pueden estar libres o restringidos dependiendo del tipo de apoyo del nodo. Por ejemplo:

- Un empotramiento restringe las tres componentes: no permite traslación en x, ni en y, ni rotación.
- Un apoyo simple restringe solo las traslaciones, pero permite la rotación libremente.

Para simplificar la notación, a partir de aquí denominaremos a los grados de libertad como GDL, y usaremos los términos GDL_{libre} y $GDL_{restringido}$ según sea necesario, para referirnos a los nodos que pueden o no desplazarse y rotar.

Bazán y Meli (2002) añaden que los grados de libertad en un sistema estructural están directamente relacionados con los desplazamientos o giros en puntos específicos. Por ejemplo, en un marco típico, los grados de libertad pueden reducirse si se tienen elementos estructurales rígidos, como diafragmas en los pisos que restringen algunos movimientos. Un marco estructural con restricciones puede pasar de tener 12 grados de libertad estáticos a una cantidad reducida si ciertas deformaciones no son significativas o están restringidas (p. 99).

En otras palabras, los grados de libertad reflejan las posibilidades de movimiento en una estructura. En un sistema estructural donde ciertos desplazamientos se consideran despreciables, se pueden reducir los grados de libertad al eliminar movimientos irrelevantes, simplificando así el análisis. Sin embargo, esto no implica que esos desplazamientos sean cero, sino que no tienen un impacto significativo en la estructura.

En el análisis estructural mediante el método de análisis matricial de estructuras, sustituimos la estructura continua real por un modelo compuesto por elementos estructurales discretos, cuyos nodos conectan dichos elementos y son los puntos donde se concentran los grados de libertad. Cada nodo en el sistema tiene un identificador único, que en este caso denotaremos como $n_1, n_2, \dots, n_n$ siguiendo un orden definido.

Cada nodo en un sistema bidimensional tiene tres grados de libertad. El cálculo de estos GDL para un nodo específico n_i se puede realizar mediante las siguientes ecuaciones generales:

Para el nodo n_i:

$$GDL_{i,1} = (n_i - 1) * 3 + 1 \quad (13)$$

$$GDL_{i,2} = (n_i - 1) * 3 + 2 \quad (14)$$

$$GDL_{i,3} = (n_i - 1) * 3 + 3 \quad (15)$$

Estas ecuaciones indican cómo se asignan los GDL a un nodo en un sistema con múltiples nodos. Los primeros dos GDL corresponden a las traslaciones en las direcciones x y y, mientras que el tercer GDL corresponde a la rotación alrededor del nodo. Esto facilita el cálculo y seguimiento de los GDL en cada nodo de la estructura durante el análisis.

Matriz de rigidez local

Rojas Rojas y Padilla Punzo (2009) definen la rigidez de un elemento estructural como la magnitud de la fuerza requerida para producir un desplazamiento unitario (p. 136). Esta definición nos permite entender que la rigidez de un elemento está directamente relacionada con su resistencia al desplazamiento bajo una carga aplicada.

Blanco Claraco et al. (2012) explican que la matriz de rigidez es una propiedad del sistema estructural, no cambia en función del estado de cargas o de condiciones de contorno a que se someta la estructura (p. 22). En otras palabras, la matriz de rigidez K de un elemento depende únicamente de sus propiedades geométricas y del material, permaneciendo constante mientras no se modifiquen estas características.

Por otro lado, Vázquez Fernández (1999) señala que la matriz de rigidez local k tiene las siguientes propiedades:

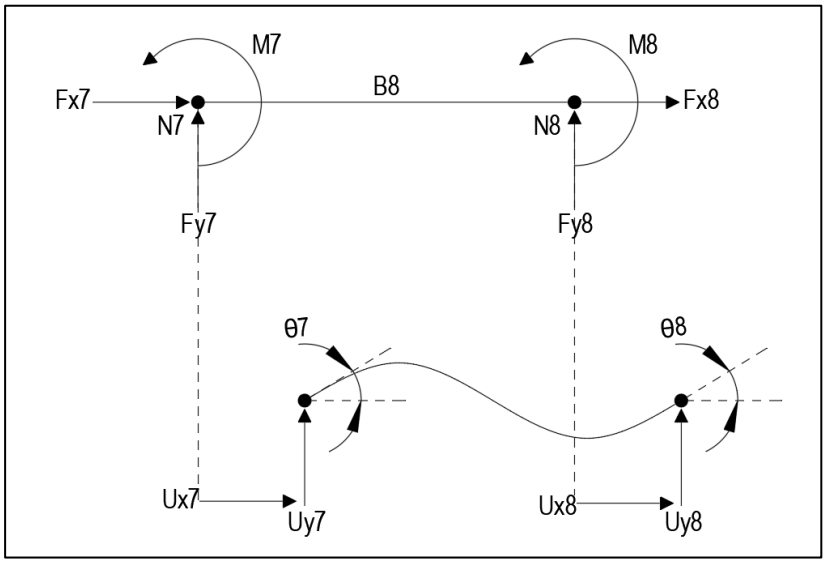
- Es una matriz cuadrada de orden 6, ya que cada nodo en el sistema bidimensional tiene tres grados de libertad (dos traslaciones y una rotación) y un elemento conecta dos nodos.
- Los elementos de la diagonal principal son positivos y no pueden ser nulos, porque el desplazamiento de un extremo de una barra en una dirección específica requiere la aplicación de una fuerza en ese mismo sentido.
- El elemento k_{ij} de la matriz representa la solicitación de orden i (esfuerzo) generada por el desplazamiento unitario de orden j (deformación).
- Es una matriz simétrica, lo que se puede demostrar mediante el teorema de Maxwell o de la reciprocidad de los trabajos (p.47).

Estas propiedades son fundamentales para entender el comportamiento de la matriz de rigidez, ya que reflejan cómo la estructura responde a los desplazamientos y fuerzas aplicadas. La simetría de la matriz garantiza que los esfuerzos y desplazamientos en un sistema estructural estén equilibrados.

Para explicar mejor estas afirmaciones, consideremos un ejemplo con el elemento $B8$, formado por los nodos N_7 y N_8, donde se aplican fuerzas externas. Cada nodo tiene tres grados de libertad: dos traslacionales $(U_x\ y\ U_y)$ y uno rotacional (θ). En este caso, analizamos los desplazamientos U_{x7} y U_{x8} y las rotaciones θ_7 y θ_8 en los nodos N_7 y N_8, respectivamente, como respuesta a las fuerzas aplicadas, como se observa en la siguiente imagen.

Figura 3

Deformada general



Nota. Adaptado de Análisis matricial de estructuras (Figura 1.14.2, p. 10), por R. Aguiar Falconi, 2004, CEINCI-ESPE.

La rigidez del elemento $B8$, como indican Rojas Rojas y Padilla Punzo (2009), es la magnitud de la fuerza necesaria para producir los desplazamientos señalados. Esto se representa matemáticamente

mediante la matriz de rigidez del elemento, que relaciona las fuerzas y desplazamientos nodales a través de la ecuación fundamental (ecuación 1).

Blanco Claraco et al. (2012) refuerzan que la matriz de rigidez es una propiedad inherente del sistema estructural, lo que implica que el elemento B_2, o cualquier otro, mantendrá su rigidez mientras sus propiedades materiales y geométricas no cambien (p. 22).

Es importante entender que la rigidez de un sistema estructural solo cambiará si se modifica la configuración de la estructura, es decir, si se agrega o retira algún elemento estructural, lo cual afectará la distribución de fuerzas y, por ende, la matriz de rigidez global del sistema. Estas afirmaciones se aplican de manera coherente en el análisis estructural realizado en esta investigación.

Rojas Rojas y Padilla Punzo (2009), Blanco Claraco et al. (2012), Quispe Panca (2015) y Vázquez Fernández (1999) señalan que los grados de libertad de un sistema se reflejan en la rigidez del elemento. Cada nodo posee tres grados de libertad, por lo que la rigidez de un elemento cubre un total de seis grados de libertad. Esto permite formular el sistema de ecuaciones que modela el comportamiento estructural a través de representaciones matriciales.

A continuación, se presentan diferentes configuraciones de elementos estructurales, según Quispe Panca (2015), que permiten visualizar cómo cambia la matriz de rigidez en función de las restricciones de los extremos:

a. **Elemento no articulado – no articulado:** Ambos extremos del elemento están rígidamente conectados, sin posibilidad de rotación, como en las estructuras de concreto reforzado. La matriz equivalente al sistema de ecuaciones de rigidez es:

$$k = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{2EI}{L} \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{2EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix} \quad (16)$$

b. **Elemento articulado - no articulado:** Un extremo del elemento permite rotación (articulado), mientras que el otro extremo está rígidamente conectado, como en algunas conexiones de vigas de acero a columnas. La matriz equivalente al sistema de ecuaciones de rigidez es:

$$k = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{3EI}{L^3} & 0 & 0 & -\frac{3EI}{L^3} & \frac{3EI}{L^2} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{3EI}{L^3} & 0 & 0 & \frac{3EI}{L^3} & -\frac{3EI}{L^2} \\ 0 & \frac{3EI}{L^2} & 0 & 0 & -\frac{3EI}{L^2} & \frac{3EI}{L} \end{bmatrix} \quad (17)$$

c. **Elemento no articulado - articulado:** Un extremo del elemento está rígidamente conectado, mientras que el otro extremo permite rotación, como en estructuras de acero donde las vigas tienen conexiones articuladas a una base rígida. La matriz equivalente al sistema de ecuaciones de rigidez es:

$$k = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{3EI}{L^3} & \frac{3EI}{L^2} & 0 & -\frac{3EI}{L^3} & 0 \\ 0 & \frac{3EI}{L^2} & \frac{3EI}{L} & 0 & -\frac{3EI}{L^2} & 0 \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{3EI}{L^3} & -\frac{3EI}{L^2} & 0 & \frac{3EI}{L^3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (18)$$

d. **Elemento articulado - articulado:** Ambos extremos del elemento permiten rotación, simulando un comportamiento más flexible, como en la estructura de una cobertura metálica o en puentes de vigas simplemente apoyadas. La matriz equivalente al sistema de ecuaciones de rigidez es:

$$k = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{19}$$

Estas configuraciones permiten analizar cómo los elementos estructurales responden a diferentes cargas y restricciones. Como se mencionó, la matriz de rigidez local de un elemento se denota por k (minúscula). En el próximo apartado se discutirá la matriz de rigidez global, que se denotará por K (mayúscula), siguiendo la convención de Quispe Panca (2015).

Matriz de global por elemento

Blanco Claraco et al. (2012) explican que las relaciones establecidas en el apartado anterior asumen que tanto las solicitaciones como los desplazamientos están dados en coordenadas locales de la barra. Para integrar toda la estructura en su conjunto, es necesario transformar estos valores a coordenadas globales, lo que permite un análisis coherente de la estructura completa.

Para realizar esta transformación, se utiliza la matriz de transformación T, que convierte las magnitudes desde el sistema local de coordenadas al global. De manera similar, la matriz T transpuesta (denotada como T^T) convierte los valores del sistema global al local. Blanco Claraco et al. (2012) definen la transformación completa de la matriz de rigidez local a global mediante la ecuación:

$$K = T * k * T^T \quad (20)$$

Sin embargo, Quispe Panca (2015) y Rojas Rojas y Padilla Punzo (2009) proponen un orden diferente para la ecuación de transformación:

$$K = T^T * k * T` \quad (21)$$

A pesar de que existe una diferencia en el orden de multiplicación entre los autores, no se trata de una discrepancia fundamental, ya que, en términos de matrices, $A * B$ no es igual a $B * A$. En esta investigación, seguiremos la convención de Quispe Panca (2015) y Rojas Rojas y Padilla Punzo (2009), donde T representa la matriz de transformación de coordenadas locales a globales, y T^T su transpuesta para la transformación inversa.

Cervera y Blanco (2004) añaden que una vez definida la relación entre las fuerzas aplicadas en los extremos de la pieza y los movimientos en los nudos, en el sistema local de cada pieza (elementos), se procede a la transformación de dichas ecuaciones al sistema global de referencia. Estos autores también explican que si el ángulo que forma el elemento con el eje de coordenadas principal x es α, este ángulo debe usarse en la matriz de transformación para convertir los componentes de x he y a las coordenadas globales x' e y'.

En resumen, la ecuación de transformación que adoptamos en esta investigación, de acuerdo con Quispe Panca y Rojas Rojas & Padilla Punzo, es la ecuación 12.

Este enfoque asegura coherencia en todo el análisis estructural y evita posibles confusiones con el orden de multiplicación de las matrices.

Ensamble de la matriz de rigidez global

Quispe Panca (2015) señala que, una vez que todas las matrices de rigidez de los elementos que conforman la estructura se han expresado en coordenadas globales, es necesario ensamblarlas en el orden apropiado para obtener la matriz de rigidez completa de la estructura K. De manera similar, Blanco Claraco et al. (2012) afirman que los elementos (barras) individuales se ensamblan para formar la matriz de rigidez global K (p. 57).

A partir de lo señalado por los autores, todos coinciden en que el siguiente paso es ensamblar las matrices de rigidez de cada barra en una única matriz global. Sin embargo, no especifican claramente cómo se debe definir esta matriz global. Aunque se refieren a ella como K, ya hemos utilizado esta notación para representar la matriz de rigidez de cada elemento individual.

Por lo tanto, en esta investigación, me referiré a la matriz de rigidez global ensamblada como K_g, a modo de preferencia personal y con el objetivo de evitar confusiones durante el desarrollo de este trabajo.

Para determinar la matriz K_g, Quispe Panca (2015) explica que es necesario numerar cada uno de los nodos y elementos de la estructura, y asignarles sus correspondientes grados de libertad. Cada grado de libertad se coloca en una fila y columna específicas dentro de la matriz. De manera similar, Blanco Claraco et al. (2012) mencionan que si numeramos cada uno de los N nudos de la estructura como $i = 1, 2, 3, n$, se puede mostrar que dicha matriz K se forma a partir de las submatrices descritas en las secciones anteriores (p. 57).

Ambos autores destacan la importancia de numerar los nodos y asignar los grados de libertad correspondientes, ya que el total de grados de libertad será el número de filas y columnas de K_g.

Arturo Tena Colunga (2007) refuerza este enfoque al señalar que la obtención de cada término de la matriz de rigidez K_g se simplifica mediante la suma de las contribuciones de cada barra en los extremos correspondientes a los desplazamientos asociados a los nodos i y j. Es decir, solo contribuye una barra si i es diferente de j, mientras que varias barras contribuyen si i es igual a j, como en el caso cuando varias barras concurren en el nodo i. Tena Colunga también menciona que esta manera de ensamblar las matrices se conoce como la regla del ensamblaje, un método descrito por Damy Ríos (1986), el cual ha sido uno de los algoritmos más utilizados en programas comerciales de análisis estructural debido a que reduce significativamente el uso de memoria. En lugar de almacenar completamente las matrices globales de rigidez de los elementos, solo se requieren las submatrices para cada barra o grupo de elementos con propiedades distintas (p. 60).

El resultado de este proceso se representa mediante la siguiente expresión para n nodos con n grados de libertad:

$$K_g = \begin{bmatrix} \sum K_{ii} & \cdots & \sum K_{ij} \\ \vdots & \ddots & \vdots \\ \sum K_{ji} & \cdots & \sum K_{jj} \end{bmatrix} \tag{22}$$

En esencia, la matriz de rigidez global K_g se obtiene sumando los valores de las matrices de rigidez de los elementos cuando coinciden en los grados de libertad correspondientes. Este proceso puede ser más sencillo cuando se trata de elementos consecutivos, aunque se vuelve más complejo en modelos estructurales avanzados donde un nodo puede pertenecer a más de dos elementos o la estructura no sea consecutiva. Este tipo de complejidad, abordada mediante la regla del ensamblaje, es uno de los aspectos clave de esta investigación, ya que permitirá validar la hipótesis planteada y explorar mejoras en la eficiencia del cálculo estructural.

Aplicación del método de rigidez

Recordemos la ecuación fundamental del método de rigidez (ecuación 1): $[f] = [K] \cdot [U]$ Aplicando la convención para la matriz ensamblada y diferenciando entre vector y matriz, tendríamos $\{F\} = [Kg] \cdot \{U\}$, donde $\left[K_g\right]$ es la matriz de rigidez global ensamblada. Al despejar $\{U\}$ se obtiene la siguiente expresión:

$$\{U\} = [Kg]^{-1} * \{F\} \tag{23}$$

Donde $\{F\}$ es el vector de fuerzas actuantes en cada nudo de la estructura modelada (Quispe Panca, 2015). Este procedimiento no se detalla por otros autores; sin embargo, todos llegan a la misma conclusión en el desarrollo del análisis.

Blanco Claraco et al. (2012) señalan que el sistema no puede resolverse de manera inmediata por estar datos e incógnitas entremezclados en los vectores $\{F\}$ y $\{U\}$ [...] Se hace necesario, por tanto, particionar el problema de forma que se tenga al menos un vector sin incógnitas (p. 66). Este autor hace referencia a la necesidad de simplificar el sistema de ecuaciones descartando los valores conocidos. Quispe Panca (2015) añade: Los valores conocidos de U en estas direcciones y los nudos mencionados son iguales a 0 (p. 61). Ambos autores se refieren a desplazamientos conocidos, es decir, cuando un grado de libertad está restringido.

Un grado de libertad se considera restringido cuando se limita su desplazamiento en traslación o rotación. Por ejemplo, en una viga empotrada, el punto de empotramiento no puede desplazarse en ninguno de sus grados de libertad debido a la naturaleza del empotramiento; por lo tanto, los desplazamientos en esos puntos son conocidos y son iguales a 0.

Si ordenamos las filas y columnas de $[K_g]$, obtendremos lo que Quispe Panca y Blanco Claraco et al. definen como la matriz de rigidez global ordenada. Esta matriz se conforma de la siguiente manera (Quispe Panca, 2015):

$$k_g = \begin{bmatrix} K_{RR} & K_{RL} \\ K_{LR} & K_{LL} \end{bmatrix} \tag{24}$$

Donde K_{RR} es la submatriz de rigidez correspondiente a los grados de libertad restringidos, K_{RL} y K_{LR} son las submatrices que relacionan los grados de libertad restringidos con los libres, y K_{LL} es la submatrices correspondiente a los grados de libertad libres.

Martín Chica (2018) introduce un enfoque alternativo para abordar la reducción de la matriz de rigidez global. En lugar de imponer las condiciones de sustentación en la matriz global completa, estas se aplican directamente en las matrices elementales de aquellas barras que tienen alguno de sus extremos en contacto con un aparato de apoyo. Estas matrices elementales sufrirán eliminaciones en sus filas y columnas, y luego se procederá al ensamblaje de forma habitual. Como señala este autor, dado que el porcentaje de estas barras es pequeño frente al total de elementos, las modificaciones serán mínimas. Este enfoque complementa la descripción dada por Quispe Panca y Blanco Claraco et al., quienes también enfatizan la necesidad de ordenamiento y reducción en la matriz para facilitar los cálculos posteriores (p. 4).

Simplificando la matriz $[K_g]$, ya es posible aplicar lo establecido en la Ec. 20, adaptado a la submatriz K_{RR}:

$$\{U\} = [K_{RR}]^{-1} * \{F\} \tag{25}$$

Fuerzas y Reacciones

Respecto al vector de fuerzas, ningún autor citado hasta el momento se refiere directamente a este aspecto en detalle. Quispe Panca (2015) menciona brevemente que las fuerzas actuantes en el sistema se colocan de forma ordenada en el vector de fuerzas (p. 62). A partir de esta explicación y lo observado en otros autores, se entiende que se hace un ensamblaje directo de las fuerzas externas.

Para los nodos, la aplicación de fuerzas es directa. Sin embargo, al considerar las cargas aplicadas sobre las barras, es necesario encontrar las fuerzas resultantes o reacciones en los extremos de las barras (en sus nodos). Esto se realiza mediante tablas de momentos de empotramientos perfectos, acorde al tipo de carga aplicada en la barra. Posteriormente, se suman las resultantes en los grados de libertad compartidos, de manera similar al ensamblaje de la matriz de rigidez global.

Es importante tener en cuenta que, mientras en los nodos las fuerzas son aplicadas directamente, en las barras las fuerzas se manifiestan como reacciones. En consecuencia, para determinar el vector de fuerzas $\{F\}$, es necesario primero determinar el vector de fuerzas en los nodos, que denominaremos $\{R1\}$, y luego el vector de fuerzas en las barras, que denominaremos $\{R2\}$. El vector de fuerza $\{F\}$ será la diferencia entre estos dos vectores:

$$\{F\} = \{R1\} - \{R2\} \tag{26}$$

Al evaluar la ecuación con los datos señalados, se logra determinar los desplazamientos de todos los grados de libertad, paso fundamental en este método. Quispe Panca (2015) indica, En seguida, y para calcular las reacciones de la estructura, se reemplaza en la ecuación

fundamental del método matricial de rigidez $\{F\} = [K_g] * \{U\}$ los valores de $\{U\}$ obtenidos (p. 63).

Retorno de resultados al sistema local

En este punto, se logró determinar con éxito los desplazamientos $\{U\}$ y las fuerzas (reacciones) $\{F\}$ de un sistema estructural. Sin embargo, es necesario retornar estos desplazamientos y fuerzas al sistema local para obtener una comprensión completa del comportamiento estructural.

Para realizar esta transformación, revisemos la expresión de la ecuación 8, donde se indica que, para la transformación de coordenadas globales a locales, es necesario multiplicar la matriz de transformación traspuesta, T^T, por la matriz o vector en cuestión. En este caso, consideramos los desplazamientos $\{U\}$ y las fuerzas $\{F\}$ para retornar a sus valores locales.

Estos datos se usarán para determinar los diagramas de esfuerzos axiales, momentos flectores y fuerzas cortantes, que nos serán útiles en los cálculos estructurales. Esta información es esencial para el diseño y análisis de la estructura, contribuyendo a la validación de la hipótesis y la optimización del diseño estructural.

Conclusiones sobre la Aplicación del Método Matricial de Rigidez

Puedo afirmar que el método matricial de rigidez es un enfoque sólido y versátil para el análisis estructural, especialmente en sistemas complejos que requieren un manejo preciso de los desplazamientos y fuerzas internas. A lo largo de esta investigación, se han integrado los conceptos clave de autores como Quispe Panca (2015), Blanco Claraco et al. (2012), y otros, con el fin de establecer una metodología coherente

que permitirá el análisis eficiente de estructuras utilizando la matriz de rigidez global ensamblada y su posterior reducción.

En mi investigación, se ha optado por seguir una convención clara para las matrices y vectores involucrados, facilitando su aplicación y garantizando la correcta interpretación de los resultados. Además, se ha abordado la necesidad de transformar las matrices locales a globales y de reducir el sistema de ecuaciones cuando ciertos grados de libertad están restringidos. Todo este proceso asegura una comprensión más profunda del comportamiento de la estructura, permitiendo no solo un análisis teórico preciso, sino también una base sólida para el desarrollo y validación de modelos computacionales.

De esta manera, mi investigación busca no solo aplicar el método matricial de rigidez, sino optimizar su implementación para abordar de manera eficiente los desafíos de diseño estructural, aportando así una solución práctica y accesible en el ámbito de la ingeniería civil. Para ello, nos ceñiremos principalmente al procedimiento planteado por Quispe Panca (2015), dado que proporciona una explicación clara y un enfoque práctico, permitiendo avanzar de manera ordenada y comprensible en el análisis y diseño estructural.

Introducción a la programación

La programación es la capacidad de darle instrucciones a una computadora para realizar tareas específicas. Según Muñoz (2023), la programación trata de la habilidad de darle instrucciones a nuestra computadora. La habilidad de programar independientemente del lenguaje de programación [...] es fundamental en el mundo de la informática y la tecnología (p. 21). Este enfoque subraya la relevancia de la programación como una habilidad esencial en el ámbito tecnológico actual. Las computadoras han jugado un papel crucial en diversos avances, desde el alunizaje hasta el progreso en campos como la medicina y la ciencia. De hecho, como lo menciona Muñoz (2023), las

computadoras han sido herramientas fundamentales para el progreso humano y lo seguirán siendo en el futuro.

En mi investigación, la programación juega un papel central, ya que busco desarrollar OpenRSE en LUA, un lenguaje de programación cuya eficiencia y accesibilidad son claves para mejorar el análisis estructural bidimensional de estructuras hiperestáticas. Es importante resaltar que el desarrollo de algoritmos es la base de cualquier programa. Como señalan Pino Herrera et al. (2020), el desarrollo de algoritmos es la base principal de un programa en cierto lenguaje de programación. [...] cuando se comprende muy bien lo que se desea resolver en un algoritmo, será más sencillo codificarlo en el lenguaje de programación de su preferencia (p. 18). Esto se relaciona directamente con la etapa inicial de los requerimientos en el desarrollo de software, donde la comprensión profunda del problema es esencial para un desarrollo eficaz.

El Lenguaje de Programación LUA

Lua es un lenguaje de programación de alto nivel que se caracteriza por su accesibilidad, lo que lo convierte en una excelente opción para quienes se inician en la programación sin experiencia previa. Según Muñoz (2023), Lua es un lenguaje de programación de alto nivel que resulta muy accesible para quienes quieren iniciarse en la programación sin experiencia previa. Sin embargo, su facilidad no significa que sea un lenguaje limitado, al contrario, posee una gran potencia que permite aprovechar al máximo las posibilidades de otros lenguajes más complejos (p. 32). Este aspecto es fundamental para mi investigación, ya que la accesibilidad y simplicidad en el uso de lenguajes de programación como LUA son clave para el desarrollo de OpenRSE.

Lua fue creado en 1993 por el Grupo de Tecnología en Computación Gráfica de la Pontificia Universidad Católica de Río de Janeiro, Brasil. Desde su creación, ha evolucionado hasta convertirse en

un lenguaje multiparadigma, lo que significa que permite la programación orientada a objetos, funcional e imperativa. Esta versatilidad lo convierte en una opción adecuada para diferentes tipos de desarrollo, sin limitarse a un único paradigma de programación (Muñoz, 2023).

Además, la versión de Lua que usaré en mi investigación será una versión adaptada de Lua 5.1, específicamente integrada por el software TI-Nspire de Texas Instruments. Como se menciona en el LUA Scripting API Reference Guide de TI-Nspire, El software TI-Nspire integra la mayoría de las bibliotecas estándar de Lua que vienen con la distribución de Lua (Texas Instruments, p. 1). Esta versión incluye la mayoría de las bibliotecas estándar de Lua, pero también presenta ciertas restricciones y funciones específicas que se detallan en su documentación. Esta adaptación será clave para asegurar la compatibilidad con el entorno de TI-Nspire, que utilizaré en mi investigación.

Ierusalimschy et al. (2008) describen Lua como un lenguaje de programación extensible diseñado para una programación procedimental general, con utilidades para la descripción de datos. Además, señalan que Lua es un lenguaje de script potente y ligero para cualquier programa que lo necesite. Este enfoque lo convierte en un lenguaje ideal para aplicaciones personalizadas, adaptadas a las necesidades del programa anfitrión que lo implemente. Lua está implementado en ANSI C, lo que asegura su ligereza y facilidad de integración con otros sistemas y entornos.

En mi investigación, esta capacidad de embebido de Lua es un punto clave, ya que permite que OpenRSE pueda interactuar con otras plataformas de manera eficiente, optimizando tanto el desarrollo como la ejecución de algoritmos de análisis estructural. Lua es también software libre, lo que facilita su accesibilidad y permite que sea utilizado sin restricciones por una amplia comunidad de desarrolladores (Ierusalimschy et al., 2008).

Características de LUA

En esta sección se describen algunas de las características más destacadas del lenguaje Lua, las cuales son fundamentales para su flexibilidad y eficiencia en diversos contextos de programación. La información presentada se basa principalmente en el Manual de referencia de Lua 5.1 (Ierusalimschy et al., 2008) y otros recursos relevantes.

a. Convenciones Léxicas

Lua sigue una serie de convenciones léxicas que determinan la forma en que se nombran las variables y se usan los operadores. A continuación, se resumen las principales:

- Los identificadores pueden incluir letras, dígitos y guiones bajos, pero no pueden comenzar con un dígito.
- Las palabras clave reservadas incluyen: and, $break$, do, $else$, $elseif$, end, $false$, for, $function$, if, in, $local$, nil, not, or, $repeat$, $return$, $then$, $true$, $until$, $while$.
- Aritméticos: $+$, $-$, $*$, $/$, $\%$, ^, #
- Comparadores: $==$, $\sim =$, $<=$, $>=$, $<$, $>$
- Asignación: $=$
- Delimitadores: $( )$, $\{ \}$, $[ ]$, ...
- Los strings pueden definirse con comillas simples o dobles, y pueden incluir secuencias de escape como $\backslash n$ (nueva línea) o $\backslash t$ (tabulación).

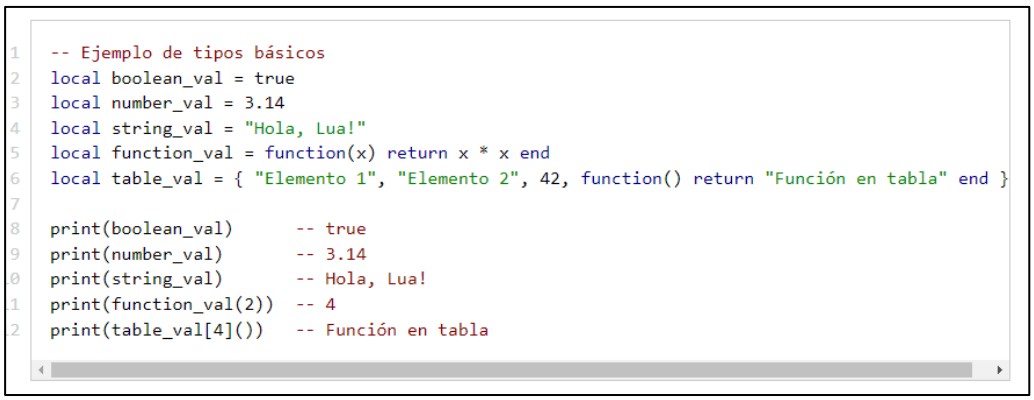
b. Valores y Tipos

Lua es un lenguaje dinámicamente tipado, lo que significa que no se especifica el tipo de las variables, sino de los valores que contienen. Los tipos básicos son:

- nil: Representa la ausencia de un valor útil.
- $boolean$: Valores lógicos, false y true.
- $number$: Números reales (de doble precisión).
- $string$: Cadena de caracteres.
- $function$: Funciones escritas en Lua o en C.
- $userdata$: Datos arbitrarios creados en C.
- $thread$: Corutinas.
- $table$: Estructuras de datos asociativas que pueden almacenar cualquier tipo de valor.

Figura 4

Ejemplo de tipos de datos en LUA



```lua
-- Ejemplo de tipos básicos
local boolean_val = true
local number_val = 3.14
local string_val = "Hola, Lua!"
local function_val = function(x) return x * x end
local table_val = { "Elemento 1", "Elemento 2", 42, function() return "Función en tabla" end }

print(boolean_val)      -- true
print(number_val)       -- 3.14
print(string_val)       -- Hola, Lua!
print(function_val(2))  -- 4
print(table_val[4]())   -- Función en tabla
```

Nota. Ejemplo de código basado en la sintaxis del Lua 5.1 reference manual (Ierusalimschy et al., 2006).

c. Variables

Las variables en Lua pueden ser globales o locales. Por defecto, las variables son globales, lo que significa que son accesibles desde cualquier parte del programa. Las variables locales se definen con la palabra clave local, y su alcance está limitado al bloque donde se declaran. Las tablas en Lua son estructuras que pueden almacenar valores de cualquier tipo y se acceden mediante índices numéricos o cadenas.

Figura 5

Ejemplos de variables globales y locales

```lua
-- Variables globales y locales
global_var = "Soy global"
local local_var = "Soy local"

-- Acceso a tablas
local tabla = { nombre = "Lua", version = 5.1 }
print(tabla["nombre"])  -- Lua
print(tabla.version)    -- 5.1
```

Nota. Ejemplo de código basado en la sintaxis del Lua 5.1 reference manual (Ierusalimschy et al., 2006).

d. **Estructuras de control**

Lua admite las estructuras de control comunes como if, $while$, $repeat$ y for. Estas permiten la ejecución condicional de bloques de código y la repetición de operaciones. A continuación, un bloque de código que muestra ejemplos de las estructuras de control más comunes en Lua:

Figura 6

Ejemplo de estructuras de control

```lua
-- Estructuras de control básicas
local x = 5

-- If-Else
if x > 0 then
    print("Positivo")
else
    print("No es positivo")
end

-- While loop
local i = 0
while i < 5 do
    print(i)
    i = i + 1
end

-- Repeat-Until loop
repeat
    print(x)
    x = x - 1
until x == 0

-- Bucle For numérico
for i = 1, 5 do
    print("Iteración " .. i)
end

-- Bucle For genérico
local tabla = {a = 1, b = 2, c = 3}
for key, value in pairs(tabla) do
    print(key .. ": " .. value)
end
```

Nota. Ejemplo de código basado en la sintaxis del Lua 5.1 reference manual (Ierusalimschy et al., 2006).

e. Concatenación de Strings

En Lua, la concatenación de strings se realiza utilizando el operador.., que permite unir varias cadenas en una sola:

Figura 7

Ejemplo de concatenación de cadenas

```lua
-- Concatenación de cadenas
local cadena1 = "Hola"
local cadena2 = "Mundo"
local mensaje = cadena1 .. ", " .. cadena2 .. "!"
print(mensaje)  -- Hola, Mundo!

```

Nota. Ejemplo de código basado en la sintaxis del Lua 5.1 reference manual (Ierusalimschy et al., 2006).

f. Definición de funciones

Las funciones en Lua se definen con la palabra clave $function$ y pueden ser anónimas o nombradas. Una característica importante de Lua es que las funciones pueden retornar múltiples valores y ser tratadas como variables.

Figura 8

Ejemplo de definición de funciones

```lua
-- Definición de una función simple
function cuadrado(x)
    return x * x
end

-- Llamada a la función
print(cuadrado(4))  -- 16

-- Funciones que retornan múltiples valores
function min_max(a, b)
    if a > b then
        return b, a
    else
        return a, b
    end
end

local menor, mayor = min_max(10, 5)
print(menor, mayor)  -- 5  10

```

Nota. Ejemplo de código basado en la sintaxis del Lua 5.1 reference manual (Ierusalimschy et al., 2006).

g. **Manejo de Errores**

Lua proporciona funciones para manejar errores sin detener la ejecución del programa. Las funciones $pcall$ y $xpcall$ son utilizadas para capturar errores en el código:

Figura 9

Ejemplo de manejo de errores

```lua
-- Manejo de errores con pcall
local status, result = pcall(function() return 1 / 0 end)  -- Error de división por cero
if not status then
    print("Error atrapado: " .. result)
end
```

Nota. Ejemplo de código basado en la sintaxis del Lua 5.1 reference manual (Ierusalimschy et al., 2006).

h. **Corutinas**

Las corutinas en Lua permiten suspender y reanudar funciones en diferentes puntos, facilitando la ejecución cooperativa:

Figura 10

Ejemplo de corutinas

```lua
-- Ejemplo de corutinas
local co = coroutine.create(function ()
    for i = 1, 3 do
        print("Corutina: " .. i)
        coroutine.yield()
    end
end)

coroutine.resume(co)  -- Corutina: 1
coroutine.resume(co)  -- Corutina: 2
coroutine.resume(co)  -- Corutina: 3
```

Nota. Ejemplo de código basado en la sintaxis del Lua 5.1 reference manual (Ierusalimschy et al., 2006).

i. Metatables y Metamethods

Las metatables permiten modificar el comportamiento de las tablas en Lua mediante metamétodos. Por ejemplo, se puede personalizar cómo una tabla maneja operaciones aritméticas:

Figura 11

Ejemplo de uso de metatables

```lua
-- Uso de metatables
local mt = {
    __add = function(t1, t2)
        return t1.valor + t2.valor
    end
}

local t1 = { valor = 10 }
local t2 = { valor = 20 }

setmetatable(t1, mt)
setmetatable(t2, mt)

local resultado = t1 + t2
print(resultado)  -- 30

```

Nota. Ejemplo de código basado en la sintaxis del *Lua 5.1 reference manual* (Ierusalimschy et al., 2006).

j. Recolección de basura

Lua maneja automáticamente la memoria mediante un sistema de recolección de basura, que elimina objetos no referenciados para liberar espacio. Esto permite al programador concentrarse en el desarrollo sin preocuparse por la gestión manual de la memoria. La función $collectgarbage()$ permite realizar ajustes en el recolector o forzar su ejecución.

¿Por qué elegir Lua y no el lenguaje más popular en la actualidad?

Python es uno de los lenguajes de programación más populares del mundo, y su utilidad en una amplia variedad de aplicaciones ha sido

reconocida por miles de programadores. Según la encuesta de Stack Overflow de 2022, Python es utilizado por un 48,07% del total de 71.547 encuestados y por un 43,51% de los desarrolladores profesionales (Mioti, 2024). Empresas como Google, NASA, CIA, Facebook, Dropbox e Instagram hacen uso intensivo de Python para una variedad de aplicaciones, especialmente en áreas como el análisis de datos, la inteligencia artificial y el desarrollo web.

Las ventajas de Python son innegables: es fácil de aprender, cuenta con una enorme cantidad de bibliotecas y módulos disponibles, y tiene una comunidad muy activa que constantemente produce recursos útiles. Sin embargo, aunque Python es una opción ideal para muchos proyectos, no siempre es la más eficiente, especialmente en situaciones donde la velocidad y la portabilidad son esenciales. Aquí es donde Lua destaca como una mejor opción en ciertos contextos.

Una de las principales ventajas de Lua sobre Python es su rendimiento. Mientras que Python ofrece una gran cantidad de abstracciones de alto nivel, estas pueden ralentizar el tiempo de ejecución. Según Hostman (2024), el código de Lua es ligero y eficiente, lo que lo hace ideal para sistemas embebidos y aplicaciones en tiempo real, donde la velocidad es crucial. En contraste, el rendimiento de Python es generalmente más lento debido a la complejidad de su estructura y sus capas adicionales de abstracción.

Python puede ser lo suficientemente rápido para muchas aplicaciones, pero en proyectos que requieren una alta eficiencia de procesamiento o deben ejecutarse en tiempo real, Lua sobresale por su velocidad y ligereza.

Además, Lua es extremadamente ligero y portable, lo que le permite funcionar en una amplia variedad de dispositivos, incluidos aquellos con recursos limitados. Su tamaño reducido y su capacidad para integrarse con otros lenguajes como C lo hacen ideal para

proyectos que requieren control directo sobre la arquitectura del sistema. Según El Blog de Python (2024), si estás trabajando en un proyecto que requiere una integración rápida y sencilla con otros lenguajes o tecnologías, Lua puede ser la mejor opción.

Python, aunque versátil, tiende a ser más pesado y requiere más recursos en comparación con Lua, lo que puede ser un inconveniente en aplicaciones que demandan un bajo consumo de memoria o necesitan ejecutarse en dispositivos con limitaciones de hardware. En estos casos, Lua se convierte en una opción más eficiente, debido a su simplicidad y bajo uso de recursos.

Otra ventaja clave de Lua es su facilidad de extensión. Lua está diseñado para integrarse fácilmente con otros lenguajes y tecnologías, lo que lo convierte en una excelente herramienta para proyectos que requieren una arquitectura extensible y adaptable. Además, Lua es software libre, lo que permite a los desarrolladores personalizar y adaptar el lenguaje según sus necesidades sin depender de costosas licencias.

Python, por otro lado, destaca en áreas como el análisis de datos y la inteligencia artificial gracias a sus amplias bibliotecas y módulos, pero su enfoque hacia la simplicidad y la abstracción lo hace menos eficiente en aplicaciones que requieren un control más directo sobre el rendimiento y los recursos del sistema. Por este motivo, en proyectos que priorizan la velocidad, la integración y la extensibilidad, Lua es una opción superior a Python.

Conclusión sobre el uso de Lua en mi investigación

En mi investigación, Lua ha sido seleccionado como el lenguaje principal debido a sus múltiples ventajas en términos de eficiencia, portabilidad y flexibilidad. Aunque Python es un lenguaje muy popular, ampliamente adoptado y con un ecosistema robusto, Lua se ajusta mejor a los requerimientos específicos de mi proyecto. Su rendimiento superior

y su capacidad para integrarse fácilmente con otros lenguajes, como C, hacen de Lua la opción ideal para aplicaciones que requieren rapidez y un uso óptimo de los recursos.

La simplicidad de Lua, combinada con su capacidad para manejar de manera eficiente sistemas embebidos y aplicaciones en tiempo real, es fundamental para alcanzar los objetivos planteados en el desarrollo de OpenRSE. Al ser un lenguaje ligero y extensible, Lua facilita la creación de un entorno de análisis estructural bidimensional eficiente, que se adapta perfectamente a las necesidades de mi investigación, donde la velocidad de procesamiento y la portabilidad del software son cruciales.

2.3. DEFINICIONES CONCEPTUALES

Accesibilidad: Facilidad de uso y disponibilidad de OpenRSE para ingenieros y estudiantes, con una interfaz amigable y accesible en diferentes plataformas.

Algoritmos de optimización: Técnicas que pueden ser implementadas en OpenRSE para mejorar la velocidad y la precisión en la resolución de los sistemas hiperestáticos.

Análisis estructural bidimensional: Evaluación de las deformaciones y tensiones en estructuras representadas en dos dimensiones, fundamental para el diseño y verificación de sistemas como pórticos o vigas.

Comparación de resultados: Evaluación de los resultados obtenidos con OpenRSE en relación con otros softwares de análisis estructural, para asegurar que sean consistentes y precisos.

Eficiencia: Capacidad de OpenRSE para ejecutar cálculos y análisis estructurales de manera rápida y optimizada, minimizando el tiempo de procesamiento y el uso de recursos computacionales.

Estructuras hiperestáticas: Sistemas estructurales con más incógnitas que ecuaciones de equilibrio, lo que requiere métodos avanzados para su resolución.

Interfaz gráfica: Herramienta que facilita la interacción del usuario con el software OpenRSE, permitiendo la visualización de modelos y resultados de forma clara e intuitiva.

LUA: Lenguaje de programación en el que está desarrollado OpenRSE, caracterizado por su ligereza, portabilidad y facilidad de integración con otros lenguajes.

Método de rigidez: Técnica que utiliza la relación entre fuerzas y desplazamientos para resolver sistemas estructurales, clave en el análisis estructural.

Metodologías de análisis: Conjunto de técnicas y procesos matemáticos y numéricos necesarios para un análisis estructural eficaz, como el método de los elementos finitos o el método de rigidez.

Precisión: Grado de exactitud de los resultados obtenidos en OpenRSE, crucial para garantizar que las estructuras analizadas sean seguras y cumplan con los requisitos de diseño.

Productividad: Incremento en la capacidad de los ingenieros para realizar análisis estructurales de manera más eficiente y accesible, contribuyendo al ahorro de tiempo y esfuerzo en comparación con otros programas como SAP2000.

Validación comparativa: Proceso en el cual los resultados de OpenRSE se contrastan con otros programas como SAP2000 y Robot Structural para asegurar precisión y robustez en el análisis.

Verificación de estructuras: Proceso de validación de los resultados obtenidos en OpenRSE, asegurando que cumplan con los estándares de diseño y las normativas estructurales.

2.4. HIPÓTESIS

2.4.1. HIPÓTESIS GENERAL

H1: Si se desarrolla OpenRSE en Lua, entonces mejorará la eficiencia y accesibilidad en el análisis estructural bidimensional de estructuras hiperestáticas en Huánuco, 2024.

H0: El desarrollo de OpenRSE en Lua no mejorará la eficiencia y accesibilidad en el análisis estructural bidimensional de estructuras hiperestáticas en Huánuco, 2024.

2.4.2. HIPÓTESIS ESPECÍFICAS

H1: Si se identifican e incorporan las características adecuadas en el desarrollo de OpenRSE en Lua, entonces se mejorará la eficiencia en el análisis de estructuras hiperestáticas en Huánuco, 2024.

H0: La identificación e incorporación de las características adecuadas en el desarrollo de OpenRSE en Lua NO mejorará la eficiencia en el análisis de estructuras hiperestáticas en Huánuco, 2024.

H1: Si se integran metodologías avanzadas de análisis estructural bidimensional de estructuras hiperestáticas en OpenRSE, entonces se garantizará un análisis más preciso y eficiente en Huánuco, 2024.

H0: La integración de metodologías avanzadas de análisis estructural bidimensional de estructuras hiperestáticas en

OpenRSE NO garantizará un análisis más preciso y eficiente en Huánuco, 2024.

H1: Si se optimiza OpenRSE en Lua mediante mejoras en su código y algoritmos, entonces se incrementará la precisión y la velocidad en la verificación de estructuras hiperestáticas bidimensionales en Huánuco, 2024.

H0: La optimización de OpenRSE en Lua mediante mejoras en su código y algoritmos NO incrementará la precisión y la velocidad en la verificación de estructuras hiperestáticas bidimensionales en Huánuco, 2024.

H1: Si se evalúa el desempeño de OpenRSE en Lua en comparación con SAP2000 y Robot Structural, entonces se demostrará que contribuye en mayor medida a la eficiencia y accesibilidad para los ingenieros en Huánuco, 2024.

H0: La evaluación del desempeño de OpenRSE en Lua en comparación con SAP2000 y Robot Structural NO demostrará que contribuye en mayor medida a la eficiencia y accesibilidad para los ingenieros en Huánuco, 2024.

2.5. VARIABLES

2.5.1. VARIABLE DEPENDIENTE

Y: Eficiencia y accesibilidad en el análisis estructural bidimensional

2.5.2. VARIABLE INDEPENDIENTE

X: Desarrollo de OpenRSE en Lua

2.6. OPERACIONALIZACIÓN DE VARIABLES (DIMENSIONES E INDICADORES)

Tabla 1

Tabla de operacionalización de variables

Variable	Definición operacional	Dimensiones	Indicadores
(X) Desarrollo de OpenRSE en Lua	Creación e implementación de OpenRSE utilizando el lenguaje Lua para analizar estructuras hiperestáticas bidimensionales.	Desarrollo del software en Lua	- Cantidad total de líneas de código desarrolladas
		Integración de metodologías de análisis estructural	- Número de tipos de estructuras soportadas
		Optimización del software	- Tiempo promedio de procesamiento por análisis
		Validación y comparación con otros softwares	- Diferencia porcentual en resultados respecto a SAP2000 y Robot Structural.
(Y) Eficiencia y accesibilidad en el análisis estructural bidimensional	Medición de la velocidad y precisión con que se realizan análisis estructurales bidimensionales de estructuras hiperestáticas utilizando OpenRSE en Lua.	Eficiencia del software	- Tiempo promedio de cálculo por modelo estructural - Tasa de error durante el análisis
		Accesibilidad del software	- Número de sistemas operativos compatibles - Tipo de licencia del software - Tamaño del archivo de instalación

Nota. Las dimensiones e indicadores han sido definidos por el autor con base en los objetivos de la presente investigación y el marco teórico desarrollado.

CAPÍTULO III
METODOLOGÍA DE LA INVESTIGACIÓN

3.1. TIPO DE INVESTIGACIÓN

3.1.1. ENFOQUE

Hernández Sampieri et al. (2014) definen la investigación como un conjunto de procesos sistemáticos, críticos y empíricos aplicados al estudio de un fenómeno o problema. A lo largo de los años, muchas corrientes de pensamiento se han polarizado en dos principales enfoques: el cuantitativo y el cualitativo. El mismo autor señala que el enfoque cuantitativo es secuencial y probatorio, donde cada etapa precede a la siguiente y no se pueden omitir pasos. Entre sus características se encuentran la medición del fenómeno y el uso de estadísticas para probar hipótesis y teorías.

En esta investigación se adopta un enfoque cuantitativo porque se centra en la recolección y análisis de datos numéricos para evaluar la eficacia y accesibilidad de la aplicación OpenRSE. Realizaremos pruebas controladas para medir el tiempo de procesamiento, la precisión de los resultados de análisis estructural y la eficiencia en el uso de recursos computacionales. Este enfoque nos permite utilizar métodos estadísticos para analizar los datos, probar hipótesis y validar teorías, asegurando un análisis objetivo y riguroso del rendimiento del software en comparación con otras herramientas existentes.

3.1.2. ALCANCE O NIVEL

Hernández Sampieri et al. (2014) subrayan la importancia de especificar el alcance de la investigación, ya que la estrategia de investigación depende de ello. Además, explican que los estudios explicativos van más allá de la simple descripción de conceptos o

fenómenos, o del establecimiento de relaciones entre conceptos; estos estudios están orientados a responder las causas de los eventos y fenómenos físicos o sociales. Este tipo de estudios busca determinar las causas de los fenómenos, generar un sentido de entendimiento y son sumamente estructurados.

El alcance de esta investigación es explicativo, ya que nuestro objetivo principal es entender y documentar el proceso de desarrollo de la aplicación OpenRSE, explicando cómo este desarrollo aborda la problemática de la falta de software accesible y eficiente para el análisis estructural en el área de la ingeniería civil. A través de un enfoque detallado y estructurado, analizaremos las decisiones de diseño y desarrollo, la implementación de algoritmos específicos y las características innovadoras de OpenRSE. Además, exploraremos el impacto de esta aplicación en el campo de la ingeniería civil, proporcionando una comprensión profunda de cómo OpenRSE puede mejorar la accesibilidad y eficiencia en el análisis estructural. La comparación del rendimiento del software con otras herramientas existentes será un aspecto secundario, complementando nuestra comprensión de su efectividad y beneficios.

3.1.3. DISEÑO

Hernández Sampieri et al. (2014) señalan que, en el enfoque cuantitativo, el investigador utiliza diseños que le permiten analizar la certeza de las hipótesis formuladas o aportar evidencias respecto a los lineamientos de la investigación. Estos diseños pueden ser experimentales o no experimentales. En los diseños experimentales, se manipulan intencionalmente una o más variables independientes para observar su efecto en las variables dependientes, controlando al mismo tiempo factores externos y asignando aleatoriamente los sujetos o unidades de análisis a los grupos experimentales y de control. Esto permite establecer relaciones de causa y efecto entre las variables estudiadas.

En esta investigación seguirá un diseño experimental, ya que manipularemos intencionalmente la variable independiente el desarrollo de OpenRSE en Lua con diferentes configuraciones o algoritmos para observar su efecto en la variable dependiente, que es la eficiencia y accesibilidad en el análisis estructural bidimensional. Estableceremos condiciones controladas y asignaremos aleatoriamente los modelos estructurales a diferentes versiones del software para aislar el efecto de las modificaciones realizadas. Realizaremos pruebas controladas para medir variables como el tiempo de procesamiento, la precisión de los resultados y la eficiencia en el uso de recursos computacionales. Este enfoque experimental nos permitirá establecer relaciones de causa y efecto entre las modificaciones en OpenRSE y las mejoras en la eficiencia y accesibilidad del análisis estructural bidimensional, validando nuestra hipótesis de manera rigurosa.

3.2. POBLACIÓN Y MUESTRA

3.2.1. POBLACIÓN

Hernández Sampieri et al. (2014) señalan que, en una investigación, la población es el conjunto de todos los posibles individuos, objetos o medidas de interés que comparten características similares. En el contexto de nuestra investigación, la población está conformada por todos los modelos matemáticos bidimensional de estructuras hiperestáticas. Esto incluye una amplia variedad de estructuras bidimensionales hiperestáticas que pueden ser objeto de análisis en el campo de la ingeniería civil, abarcando diversas configuraciones, materiales y condiciones de carga.

Debido a la amplitud y diversidad de esta población, trabajar con todos los posibles modelos es impracticable. Además, determinar una población y una muestra correspondiente mediante cálculos probabilísticos resulta inviable, ya que no contamos con una población definida en términos estadísticos tradicionales.

3.2.2. MUESTRA

Hernández Sampieri et al. (2014), al abordar si en una investigación siempre se trabaja con muestras, señalan que no siempre, aunque en la mayoría de las situaciones sí se realiza el estudio en una muestra. Como se explicó anteriormente en esta investigación, determinar una población y una muestra correspondiente mediante cálculos probabilísticos resulta inviable.

En consecuencia, esta investigación se enmarca en el muestreo no probabilístico o dirigido. Hemos seleccionado intencionalmente una muestra de modelos específicos de estructuras bidimensionales hiperestáticas para guiar y validar el desarrollo de la aplicación OpenRSE. Esta muestra no pretende ser estadísticamente representativa de toda la población, sino que se elige deliberadamente para asegurar que el desarrollo del software sea robusto y abarque una gama significativa de casos relevantes.

Los modelos seleccionados en nuestra muestra incluyen:

- Marcos rígidos con diversos números de pisos y vanos.
- Cerchas planas con distintas configuraciones y cargas.
- Pórticos sometidos a cargas verticales y laterales.
- Estructuras con materiales y secciones variadas.

Al utilizar este muestreo dirigido, nos enfocamos en modelos clave que nos permiten profundizar en el análisis y obtener datos relevantes para validar nuestra hipótesis. Esto contribuye a demostrar la efectividad de OpenRSE en mejorar la eficiencia y accesibilidad en el análisis estructural bidimensional de estructuras hiperestáticas.

3.3. TÉCNICAS E INSTRUMENTO DE RECOLECCIÓN DE DATOS.

3.3.1. PARA LA RECOLECCIÓN DE DATOS

Para alcanzar los objetivos planteados, se emplearán las siguientes técnicas:

Observación Sistemática

La observación sistemática es una técnica de recolección de datos que implica el registro planificado y estructurado de comportamientos y eventos tal como ocurren en su contexto natural (Hernández Sampieri et al., 2014). Esta técnica es especialmente útil para obtener datos directos y detallados sobre fenómenos específicos.

En esta investigación, se utilizará la observación sistemática para registrar detalladamente el comportamiento y rendimiento de la aplicación OpenRSE desarrollada en LUA durante las pruebas. Se observarán indicadores clave como:

- **Desplazamientos y Reacciones en Nodos:** Medición de los desplazamientos y fuerzas en cada nodo del sistema estructural.
- **Momentos Flectores, Esfuerzos Cortantes y Axiales en Barras:** Evaluación de los esfuerzos máximos en cada elemento estructural.
- **Tiempo de Procesamiento:** Registro del tiempo requerido por la aplicación para realizar cálculos y generar resultados.

La observación permitirá obtener datos empíricos sobre el desempeño real del software en diferentes escenarios de prueba, proporcionando información valiosa para su evaluación y mejora.

Pruebas estandarizadas

Las pruebas estandarizadas son instrumentos diseñados para medir variables específicas de manera objetiva y consistente, facilitando la comparación de resultados (Hernández Sampieri et al., 2014). Estas pruebas se basan en procedimientos y criterios uniformes, lo que garantiza la confiabilidad y validez de las mediciones.

En el contexto de esta investigación, se aplicarán pruebas estandarizadas para evaluar la precisión y eficiencia de OpenRSE en comparación con otros programas especializados como SAP2000 y Robot Structural Analysis. Se utilizarán modelos matemáticos y casos de estudio previamente validados que servirán como referencia para:

- **Comparación de Resultados:** Evaluar la concordancia entre los resultados de OpenRSE y los obtenidos con los programas de referencia.
- **Análisis de Eficiencia:** Comparar el tiempo de procesamiento y el uso de recursos entre las diferentes aplicaciones.
- **Validación de Conformidad:** Verificar que OpenRSE cumple con las normativas y estándares técnicos aplicables en el análisis estructural.

3.3.2. PARA LA PRESENTACIÓN DE DATOS

Instrumentos metodológicos

Se empleará un formato estructurado único para el registro preciso y consistente de los resultados obtenidos durante las pruebas. Este formato, detallado en el Anexo 2, permite capturar simultáneamente la información de los nodos y las barras del sistema estructural analizado. El formato contiene las siguientes características:

- Desplazamientos y Reacciones en Nodos: Registro de los valores obtenidos para cada nodo (n1, n2, n3, etc.).

- Momentos Flectores, Esfuerzos Cortantes y Axiales en Barras: Documentación de los esfuerzos máximos en cada barra (b1, b2, b3, etc.).
- Identificación de la Fuente de Datos: Indicación clara de si los resultados corresponden a OpenRSE, SAP2000 o Robot Structural Analysis.

Instrumentos Físicos

Para llevar a cabo las pruebas y el desarrollo de la aplicación OpenRSE, se utilizarán varios instrumentos físicos esenciales:

- **Computadora Portátil:** Se empleará un equipo con especificaciones técnicas adecuadas que permitirá desarrollar y ejecutar la aplicación OpenRSE, así como instalar y utilizar los programas de referencia SAP2000 y Robot Structural Analysis. La computadora será fundamental para realizar las pruebas de rendimiento y analizar los resultados obtenidos en un entorno controlado.
- **Software TI-Nspire™ CX CAS Student (versión 5.1):** Este entorno de desarrollo integrado (IDE) se utilizará para programar en LUA y ejecutar la aplicación OpenRSE. Permitirá visualizar directamente los resultados de las pruebas en la consola y facilitará el proceso de desarrollo y depuración del código.
- **Dispositivo Handheld TI-Nspire™ CX CAS:** Se empleará esta calculadora gráfica avanzada para probar la aplicación en un entorno de hardware específico. Esto permitirá evaluar el rendimiento y funcionalidad de OpenRSE en condiciones de uso reales y portátiles, verificando su operatividad más allá del entorno de la computadora.
- **Programas Especializados:** Se utilizarán licencias (educativas) de SAP2000 y Robot Structural Analysis, programas reconocidos en el campo del análisis estructural, que servirán como referencias para las pruebas comparativas y para validar y contrastar los resultados obtenidos con OpenRSE.

3.3.3. PARA EL ANÁLISIS E INTERPRETACIÓN DE LOS DATOS

Organización y Registro de Datos en Excel

La información recopilada durante las pruebas realizadas con OpenRSE, SAP2000 y Robot Structural Analysis será organizada y registrada meticulosamente utilizando el software Microsoft Excel. Este programa permitirá estructurar los datos de manera clara y ordenada, facilitando su manejo y posterior análisis. El formato estructurado único, detallado en el Anexo 2, integrará en un solo documento los resultados correspondientes a:

- **Desplazamientos y Reacciones en Nodos:** Se registrarán los valores obtenidos para cada nodo del sistema estructural, identificados como n1, n2, n3, etc.
- **Momentos Flectores, Esfuerzos Cortantes y Axiales en Barras: Se** documentarán los esfuerzos máximos en cada barra, identificadas como b1, b2, b3, etc.

Al consolidar los datos de los tres programas en un único formato, se facilitará la comparación directa y eficiente entre los resultados obtenidos. Excel ofrece herramientas avanzadas que permitirán realizar cálculos automáticos, generar gráficos comparativos y aplicar filtros, contribuyendo así a un análisis más profundo y detallado de la información.

Verificación y Validación de Resultados

Una vez organizada la información, se procederá a la verificación y validación de los resultados obtenidos. Este proceso implicará una revisión exhaustiva de los datos para asegurar su consistencia y exactitud. Se comprobará que los resultados sean coherentes entre sí y que cumplan con las leyes fundamentales de la mecánica estructural.

Para validar la precisión de OpenRSE, se realizará una comparación detallada con los resultados proporcionados por SAP2000 y Robot Structural Analysis, programas reconocidos en el campo del análisis estructural. Se establecerá un margen de error aceptable, basado en estándares técnicos y académicos, y se verificará que los resultados de OpenRSE se encuentren dentro de este rango. En caso de identificar discrepancias significativas, se investigarán las posibles causas, que podrían deberse a diferencias en los métodos de cálculo, configuraciones de los programas o errores en los datos de entrada. Si es necesario, se realizarán ajustes en el modelo o en los algoritmos utilizados en OpenRSE para corregir dichas discrepancias.

Evaluación del Proceso de Desarrollo

Además del análisis de los resultados numéricos, se llevará a cabo una evaluación detallada del proceso de desarrollo de la aplicación OpenRSE. Se documentarán todas las etapas del desarrollo, desde la planificación hasta la implementación, incluyendo:

- **Pasos y Metodologías Empleadas:** Se describirán las estrategias y enfoques adoptados durante el desarrollo del software, así como las herramientas y recursos utilizados.
- **Dificultades Encontradas:** Se identificarán los desafíos técnicos y conceptuales enfrentados, como problemas de compatibilidad, limitaciones del lenguaje LUA o complejidades en la implementación de algoritmos.
- **Soluciones Implementadas:** Se explicarán las estrategias adoptadas para superar las dificultades, detallando cómo se resolvieron los problemas y qué modificaciones se realizaron en el código o en el diseño.

Interpretación de Datos

La interpretación de los datos es esencial para extraer conclusiones relevantes y significativas de la investigación. Se analizarán los resultados obtenidos en el contexto del marco teórico y los objetivos planteados inicialmente. Esto implicará:

- **Análisis de Cumplimiento de Objetivos:** Se evaluará cómo los hallazgos responden a las preguntas de investigación y si se alcanzaron los objetivos establecidos.
- **Identificación de Patrones y Tendencias:** Se buscarán regularidades y comportamientos recurrentes en los datos que puedan indicar fortalezas o debilidades de OpenRSE.
- **Contextualización en el Marco Teórico:** Se relacionarán los resultados con los conceptos y teorías previamente estudiados, analizando las implicaciones y relevancia de los hallazgos.

Presentación de Resultados

Finalmente, se elaborarán informes detallados que presentarán de manera clara y comprensible los resultados de la investigación. Estos informes incluirán:

- **Gráficos y Tablas:** Se utilizarán visualizaciones que faciliten la comprensión y comparación de los datos entre OpenRSE, SAP2000 y Robot Structural Analysis.
- **Descripciones Narrativas:** Se redactarán explicaciones detalladas que contextualicen los hallazgos, resaltando los aspectos más relevantes y significativos.
- **Anexos:** Se incluirá el Anexo 2 con el formato de registro de resultados, permitiendo la transparencia y verificación de la información presentada.

La presentación se enfocará en asegurar que los resultados sean accesibles tanto para expertos en el área como para lectores con conocimientos generales en ingeniería estructural. Se prestará especial atención a la claridad y precisión en la exposición de la información, facilitando su análisis y comprensión por parte de los interesados.

CAPÍTULO IV
RESULTADOS

4.1. PROCESAMIENTO DE DATOS

4.1.1. CARACTERÍSTICAS DEL DESARROLLO Y DISEÑO DEL SOFTWARE

El desarrollo de OpenRSE en el lenguaje Lua, sobre la plataforma TI-Nspire CX CAS, ha sido estructurado de manera modular con base en principios de programación orientada a objetos. La arquitectura general del programa se compone de clases principales (que definen los componentes funcionales del sistema) y controladores de eventos (que permiten la interacción entre el usuario y el software a través del teclado y el entorno gráfico del dispositivo).

En total, el sistema alcanza un volumen de 3541 líneas de código, lo que refleja el nivel de complejidad y funcionalidad integrados en el software.

Tabla 2

Estructura de Clases

Clase	Función Principal	Descripción Técnica
grafico	Visualización del modelo estructural	Permite representar gráficamente barras, nodos, apoyos, cargas, y deformadas. Recibe los datos de entrada y los transforma en un modelo visual interactivo.
entrada	Gestión de datos de entrada	Contiene los cuadros para que el usuario introduzca datos numéricos del modelo (coordenadas, longitudes, cargas, apoyos, etc.).
menu	Navegación e interacción	Controla el cambio entre ventanas o secciones del software. Permite acceder a funciones como crear modelo, asignar cargas, correr análisis, visualizar resultados, etc.

analizar	Ejecución del análisis estructural	Recolecta todos los datos del modelo, ejecuta el análisis estructural, y procesa los resultados de esfuerzos, reacciones y desplazamientos.
modelo	Almacenamiento de resultados	Recibe y organiza los resultados del análisis (reacciones, fuerzas internas, deformaciones), y los envía a grafico para su visualización.

Nota. La tabla describe la arquitectura modular y las clases principales diseñadas para el software OpenRSE, siguiendo un enfoque de programación orientada a objetos.

Los controladores de eventos son funciones predefinidas por la plataforma que se activan automáticamente cuando el usuario realiza una acción, como presionar una tecla o iniciar el programa. En OpenRSE, estos controladores permiten una interacción fluida y en tiempo real con el usuario.

Tabla 3

Controladores de eventos del entorno TI-Nspire CX CAS

Controlador	Función	Aplicación en OPENRSE
on.construction()	Inicialización del sistema	Se ejecuta al iniciar el programa. Aquí se configuran restricciones iniciales (por ejemplo, control de acceso).
on.activate()	Preparación del entorno	Se definen elementos básicos como el menú principal, y se crean las instancias de las clases (menu, entrada, grafico, etc.).
on.paint()	Dibujo gráfico en pantalla	Permite mostrar gráficamente el modelo estructural. Llama a los métodos de grafico según lo indique el menu.
on.tabKey()	Navegación con tecla Tab	Permite moverse entre campos o elementos. Usado por menu para cambiar entre secciones del software.
on.enterKey()	Confirmación con tecla Enter	Confirma acciones. Utilizado por entrada para registrar datos, y por menu para ejecutar funciones.
on.arrowDown() / on.arrowUp() /	Movimiento con teclas direccionales	Usados por entrada y menu para moverse entre campos y ventanas.

on.arrowLeft() / on.arrowRight()		
on.charIn()	Ingreso de caracteres	Permite escribir datos en los cuadros de entrada. Específico para la clase entrada.
on.backspaceKey()	Borrar texto	Permite editar los datos ingresados por el usuario. También es exclusivo de entrada.

Nota. Los controladores de eventos presentados son exclusivos de la API de desarrollo LUA del sistema TI-Nspire CX CAS y no forman parte del lenguaje Lua en su sintaxis estándar.

Esta estructura organizada en módulos y controladores permite a OpenRSE operar como un software interactivo, accesible y funcional, orientado específicamente al análisis estructural de modelos bidimensionales hiperestáticos. Gracias a la separación clara de funciones y a la coordinación entre eventos y clases, el sistema responde en tiempo real a las acciones del usuario, presentando una interfaz gráfica eficiente y fácil de usar, sin requerir que el usuario final tenga conocimientos de programación.

Además, el diseño modular del código no solo favorece la eficiencia operativa, sino también la accesibilidad a nivel de desarrollo, ya que facilita la comprensión y modificación del código fuente. Esto permite que, en futuras investigaciones o versiones extendidas del programa, se puedan incorporar nuevos métodos de cálculo o funcionalidades adicionales de manera estructurada y clara.

Organización Interna de la Clase analizar

La clase analizar es el componente central de OpenRSE en lo que respecta a la ejecución del análisis estructural. Toda la lógica computacional implementada en esta clase se basa en el método del análisis matricial de estructuras bidimensionales, el cual es ampliamente reconocido en ingeniería estructural por su capacidad para resolver

estructuras hiperestáticas mediante un enfoque sistemático, algebraico y adaptable a medios computacionales.

Este método, aplicado en estructuras planas, considera el comportamiento de cada elemento estructural a través de su matriz de rigidez, la cual se ensambla a una matriz global que representa el sistema completo. Posteriormente, se aplica la resolución del sistema de ecuaciones para obtener desplazamientos, reacciones y esfuerzos internos.

En OpenRSE, la implementación de este método se realiza mediante una organización altamente estructurada de código, distribuida en funciones específicas que ejecutan cada etapa del procedimiento. La clase está compuesta por 683 líneas de código, lo que da cuenta del nivel de detalle y capacidad analítica incorporada. Esta división en métodos no solo favorece la eficiencia operativa, sino también la claridad y trazabilidad del proceso, permitiendo que cada cálculo sea identificado y comprendido en su contexto.

A diferencia de los programas comerciales, donde los procesos internos son generalmente opacos (caja negra), en OpenRSE los cálculos están expuestos de forma explícita. Esto brinda una ventaja significativa en entornos académicos e investigativos, ya que los usuarios pueden verificar, adaptar o extender los procedimientos con facilidad, manteniendo siempre la referencia directa a los fundamentos estructurales.

Además, al seguir la lógica natural del análisis manual estructural, la clase analizar resulta fácilmente interpretable incluso para profesionales de la ingeniería civil sin formación avanzada en programación. Su diseño modular permite visualizar el paso a paso del análisis, desde la geometría inicial hasta los resultados de esfuerzos internos.

Tabla 4

Métodos Principales de la Clase analizar

Etapa del Cálculo	Método (función)	Descripción del Proceso
Cálculo preliminar	calcular_longitud_angulo_barra()	Determina la geometría básica de cada barra: longitud y ángulo.
Matrices locales y transformaciones	calcular_k_local(), calcular_t(), calcular_t_traspuesta()	Calcula las matrices de rigidez local y sus transformaciones al sistema global.
Matriz global del sistema	calcular_k_global(), calcular_k_global_ensamblada()	Ensambla la matriz de rigidez total de la estructura.
Clasificación de grados de libertad	clasificar_gdl()	Identifica grados de libertad libres, restringidos y articulados.
Ordenamiento y partición	ordenar_k_global_ensamblada()	Reorganiza la matriz para facilitar el cálculo por submatrices.
Cargas nodales	ensamblar_R1()	Ensambla el vector de cargas nodales externas.
Cargas distribuidas	ensamblar_R2()	Calcula los efectos equivalentes de cargas distribuidas sobre cada barra.
Vector resultante	calcular_R()	Resta cargas distribuidas a cargas nodales para obtener el vector de equilibrio.
Reducción del sistema	reducir_R()	Extrae el sistema correspondiente a los grados de libertad libres.
Solución del sistema	calcular_vector_desplzamiento_gdl_libre()	Resuelve el sistema de ecuaciones utilizando la matriz inversa.
Reordenamiento de desplazamientos	calcular_vector_desplazamiento_local(), calcular_vector_desplazamiento_global()	Reorganiza los desplazamientos según nodos y grados de libertad.
Reacciones	calcular_vector_reaccion_local(), calcular_vector_reaccion_global()	Calcula las reacciones estructurales en apoyos o conexiones.
Esfuerzos internos	calcular_vector_esfuerzos_barra()	Determina los esfuerzos internos (fuerza axial, cortante y momento).

Resultados gráficos	calcular_puntos_criticos_cortante(), calcular_puntos_criticos_momento()	Determina posiciones críticas para la representación de diagramas.

Nota. Todos los métodos descritos corresponden a la implementación computacional del análisis matricial de estructuras bidimensionales.

Cada uno de estos métodos opera con parámetros claramente definidos, lo que permite verificar fácilmente qué información entra y sale en cada etapa. Esto representa una ventaja importante frente a software especializado, donde el proceso de cálculo se encuentra oculto para el usuario final.

Gracias a esta organización, el sistema no solo permite realizar un análisis estructural completo, sino que ofrece un alto nivel de transparencia, trazabilidad y adaptabilidad para futuras extensiones, como nuevos tipos de cargas, condiciones de borde o refinamientos en la modelación.

Interfaz gráfica de usuario y flujo de navegación

Como parte fundamental del desarrollo de OpenRSE, se implementó una interfaz gráfica pensada para facilitar la experiencia del usuario en todas las etapas del análisis estructural. Esta interfaz se estructura en distintas ventanas que permiten el ingreso de datos, la visualización del modelo y la interpretación de los resultados, sin necesidad de que el usuario tenga conocimientos de programación o manipulación de matrices.

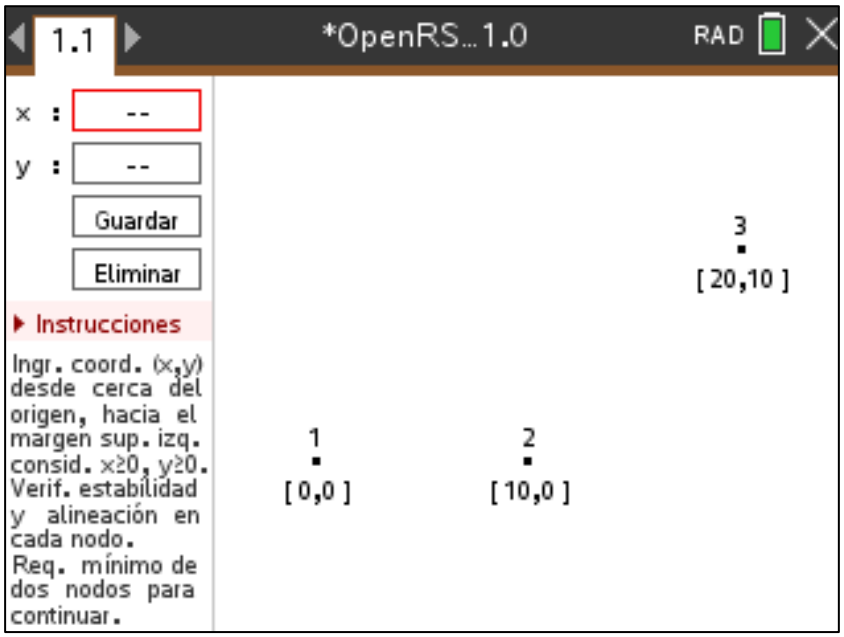
La navegación entre estas secciones se realiza de forma guiada y jerárquica, a través del menú principal del software. A continuación, se detallan las ventanas que componen la interfaz, agrupadas por función y orden lógico de uso. Cada una de ellas puede ser acompañada por su respectiva imagen o captura en el documento final.

Ventanas de ingreso de datos:

Este conjunto de ventanas constituye el primer bloque del flujo de uso de OPENRSE. Aquí el usuario ingresa la información estructural necesaria para construir el modelo: nodos, elementos, apoyos, condiciones de frontera, propiedades físicas y cargas. Estas ventanas han sido diseñadas para ser intuitivas, guiando paso a paso el proceso de modelado sin requerir conocimientos técnicos complejos. La validación de datos se realiza automáticamente para evitar errores de entrada.

Figura 12

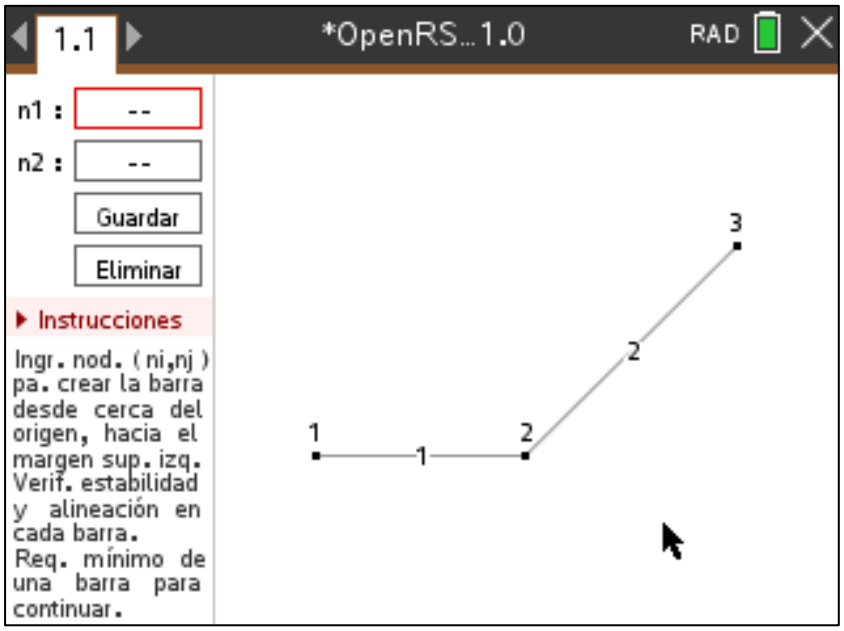
Ventana de ingreso de nodos del modelo estructural



Nota. El sistema permite el registro de nodos del modelo estructural mediante coordenadas cartesianas en el primer cuadrante (x+, y+).

Figura 13

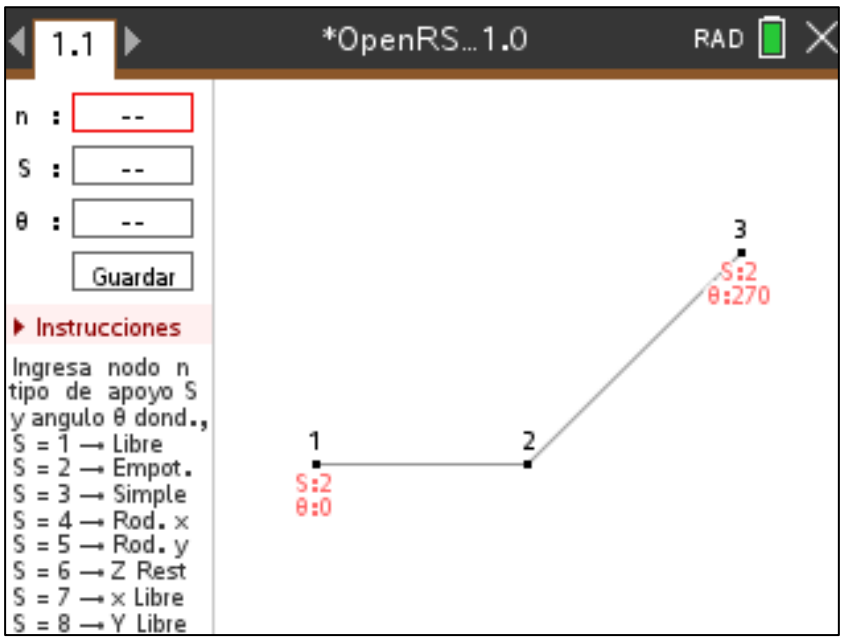
Ventana de asignación de barras



Nota. Asocia barras a los nodos previamente definidos, indicando los nodos de inicio y fin (n1, n2).

Figura 14

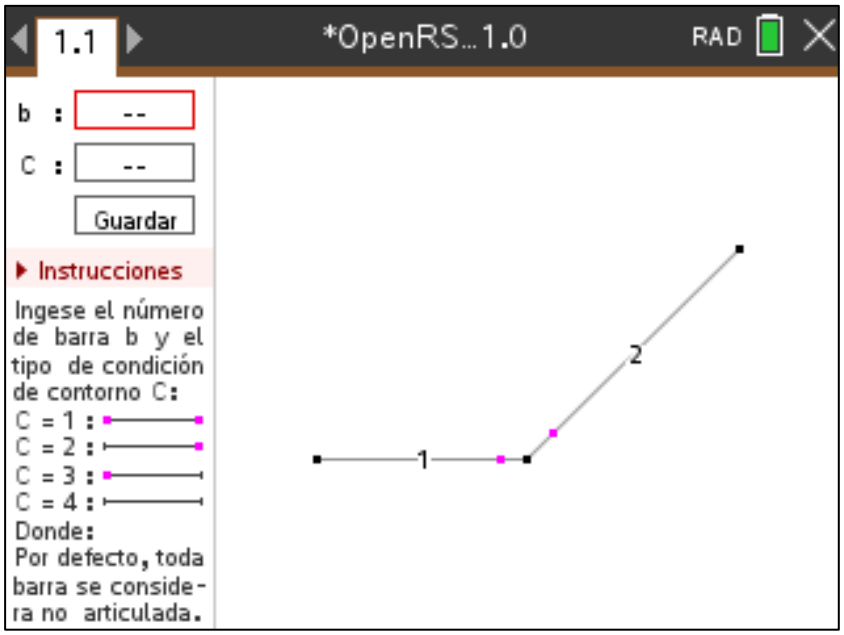
Ventana de asignación de apoyos y ángulos de rotación



Nota. Selecciona el tipo de apoyo entre 8 configuraciones posibles de grados de libertad, incluyendo ángulo de orientación.

Figura 15

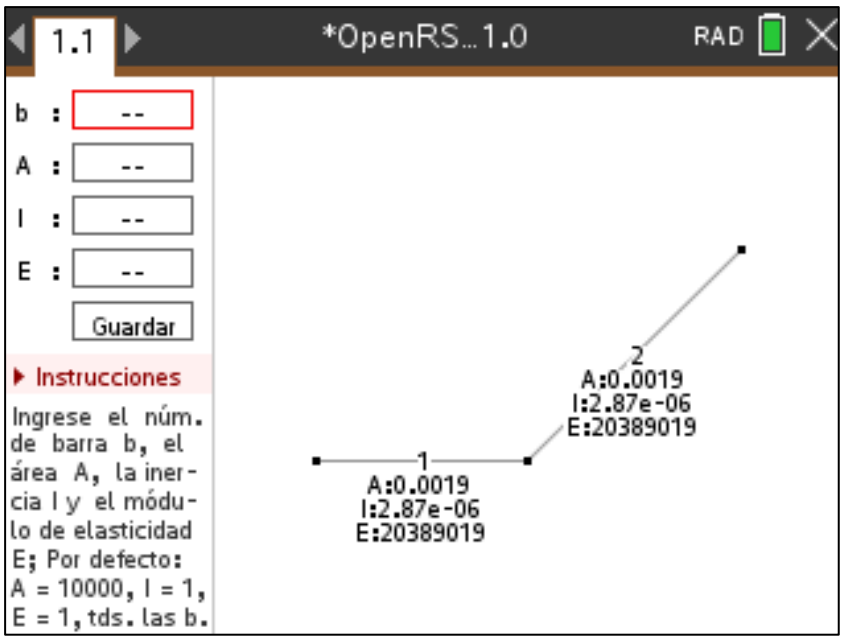
Ventana de asignación de condiciones de frontera



Nota. Permite definir si los extremos de las barras están empotrados o articulados (4 tipos posibles)

Figura 16

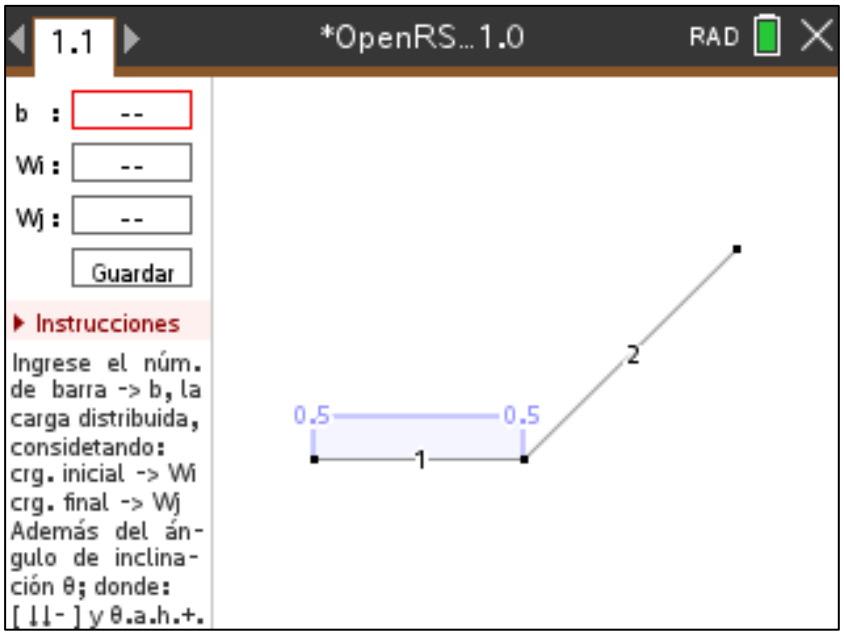
Ventana para definir las propiedades físicas y sección



Nota. Se ingresan valores de área, inercia y módulo de elasticidad para cada barra.

Figura 17

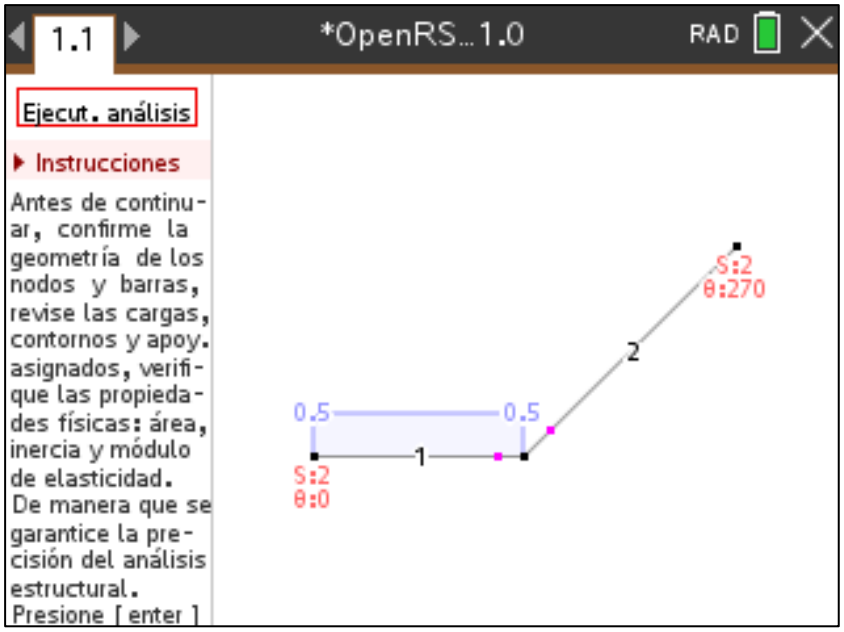
Ventana de asignación de cargas distribuidas o puntuales



Nota. Permite aplicar cargas distribuidas en barras, indicando el número de barra y la carga al inicio del tramo Wi y final del tramo Wj.

Figura 18

Ventana de resumen de datos ingresados



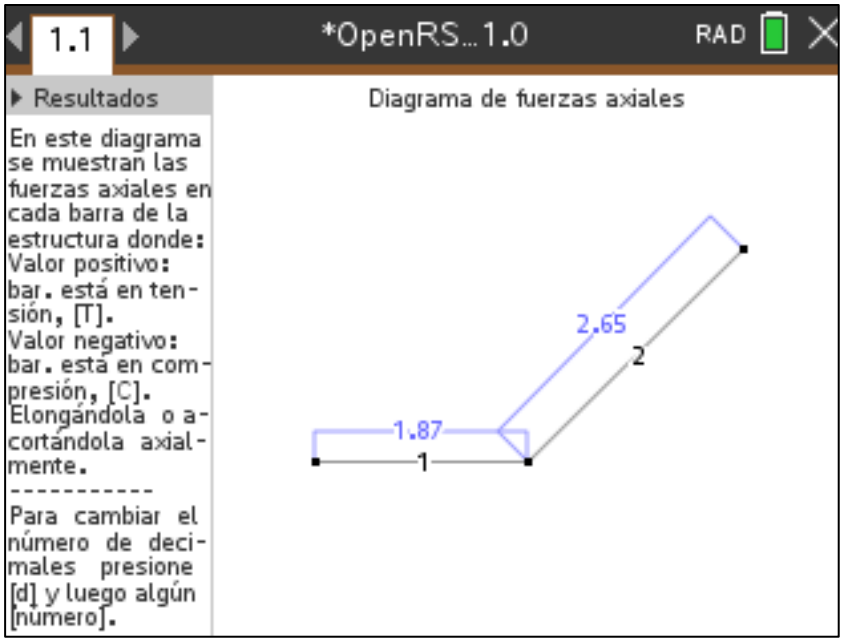
Nota. Se presenta un resumen completo de todos los datos asignados en las ventanas previas, danto paso a análisis del modelo estructural.

Ventanas de visualización de resultados gráficos:

Una vez realizado el análisis estructural, OPENRSE permite visualizar los resultados de forma gráfica. Este grupo de ventanas muestra los diferentes tipos de esfuerzos internos y respuestas estructurales, como desplazamientos o reacciones. La interfaz gráfica presenta esta información de forma clara y ordenada, permitiendo al usuario interpretar fácilmente el comportamiento del modelo sin necesidad de análisis matemático adicional.

Figura 19

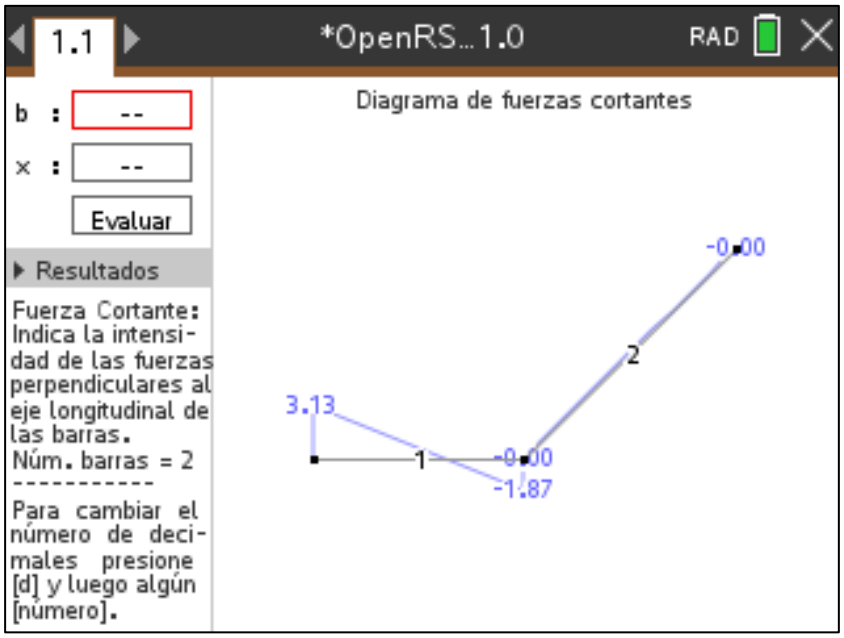
Ventana de los esfuerzos axiales del modelo



Nota. Se presenta del diagrama de esfuerzos axiales generados en el análisis estructural de manera general, se indica positivo a tracción y negativo a compresión.

Figura 20

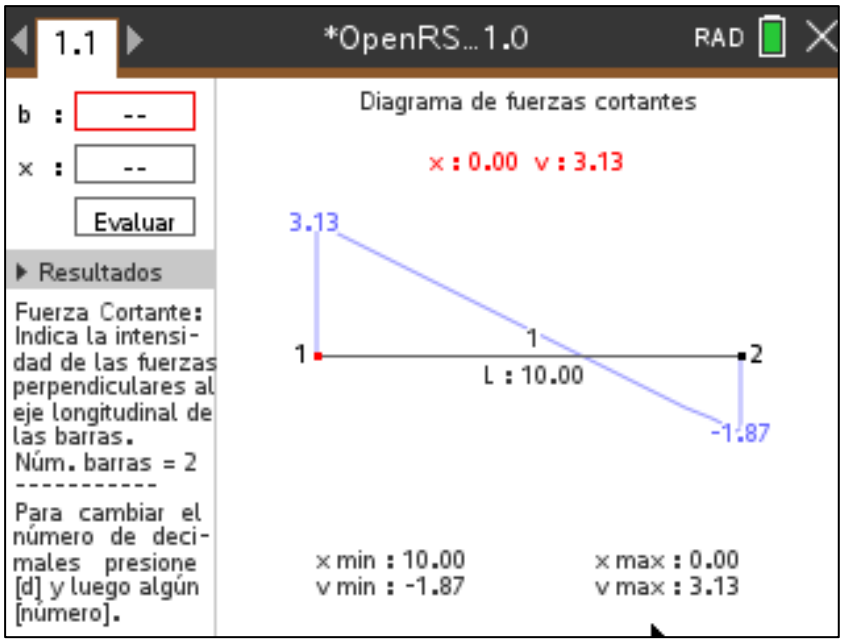
Ventana de comportamiento cortante del modelo



Nota. Se presenta los esfuerzos cortantes generados en el análisis estructural de manera general.

Figura 21

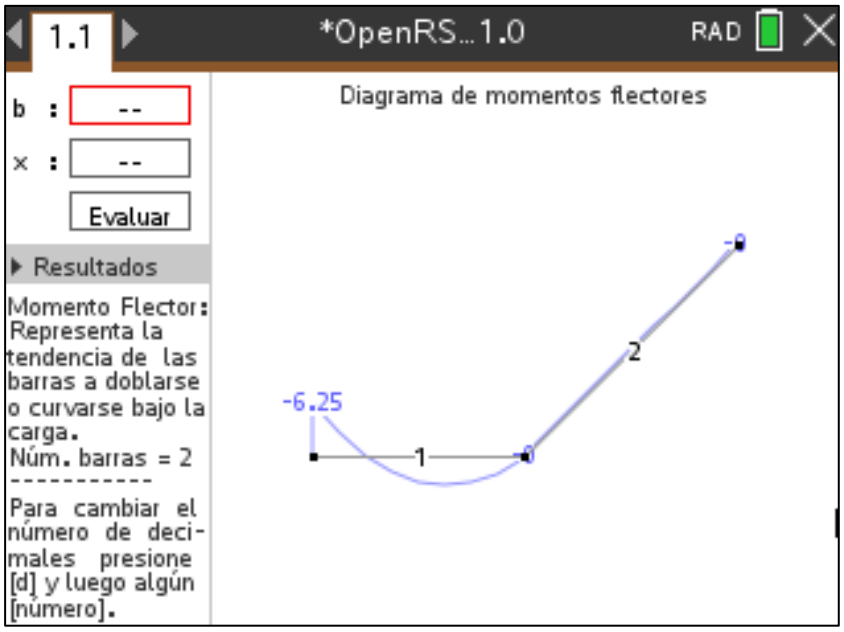
Ventana de comportamiento cortante por barra



Nota. Se indican valores máximos y mínimos del esfuerzo cortante en la barra seleccionada, así como el valor en un punto especifico.

Figura 22

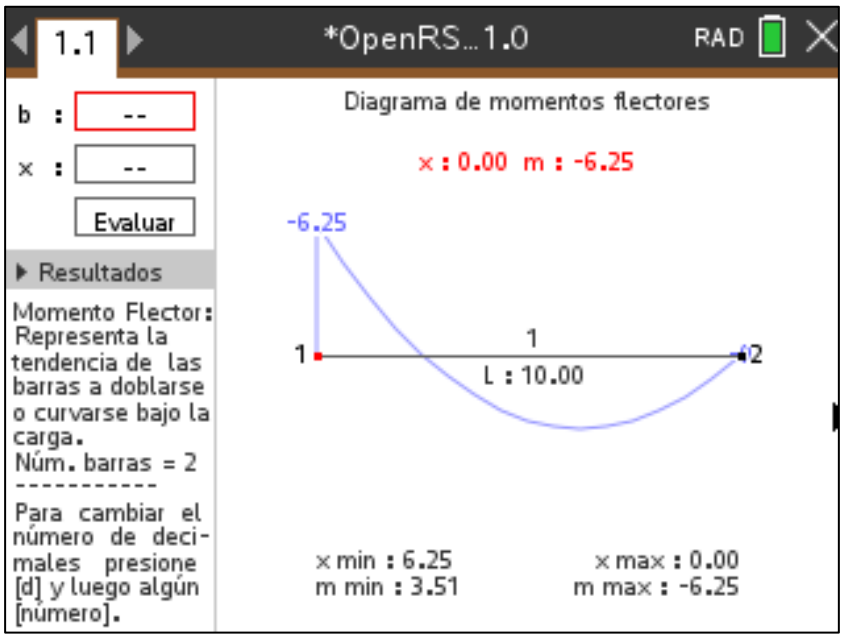
Ventana del comportamiento flector del modelo



Nota. Se presenta los momentos flectores generados en el análisis estructural.

Figura 23

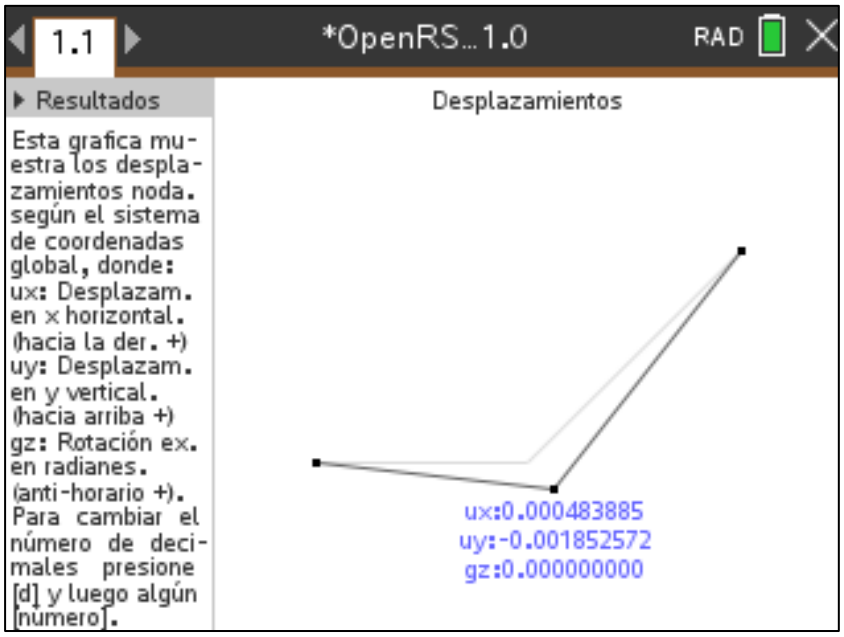
Ventana del comportamiento flector por barra



Nota. Se indican valores máximos y mínimos del momento flector en la barra seleccionada, así como el valor en un punto especifico.

Figura 24

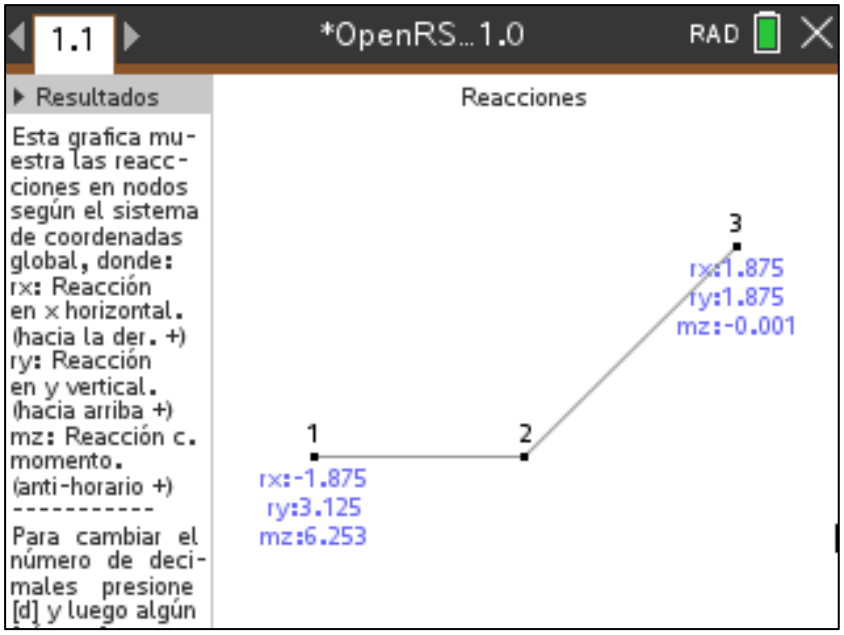
Ventana del desplazamiento general del modelo



Nota. Se presentan los desplazamientos en la dirección ux y uy, asi como la rotación de los nodos del modelo analizado.

Figura 25

Ventana de resultantes (reacciones) del modelo



Nota. Se presentan las reacciones den las direcciones rx y ry, asi como momento presente en los apoyos asignados.

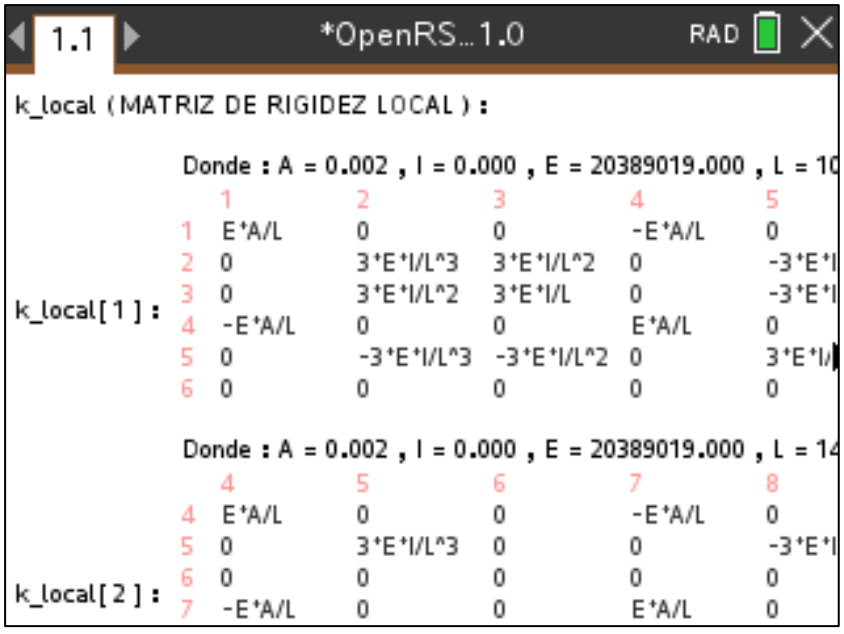
Ventanas de visualización numérica de matrices y vectores:

Este bloque final de la interfaz permite acceder a los resultados del análisis en formato numérico. Las matrices de rigidez y los vectores de fuerza, desplazamiento y reacción se muestran con claridad, ofreciendo transparencia sobre el proceso de cálculo. Esto es especialmente útil en contextos académicos, donde se requiere validar los procedimientos internos.

A diferencia de muchos programas comerciales, que solo presentan resultados finales sin revelar su desarrollo, OpenRSE permite examinar cada etapa del análisis. Esta apertura facilita la interpretación, comparación y estudio detallado, haciendo al software más accesible, didáctico y eficiente.

Figura 26

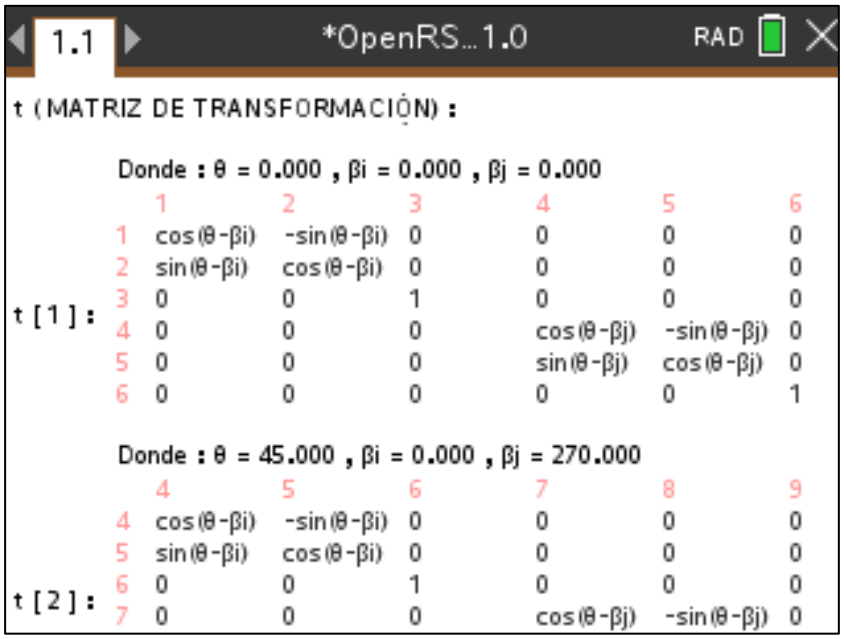
Ventana de matrices de rigidez local



Nota. Muestra la matriz K local generada para cada barra según sus condiciones de frontera.

Figura 27

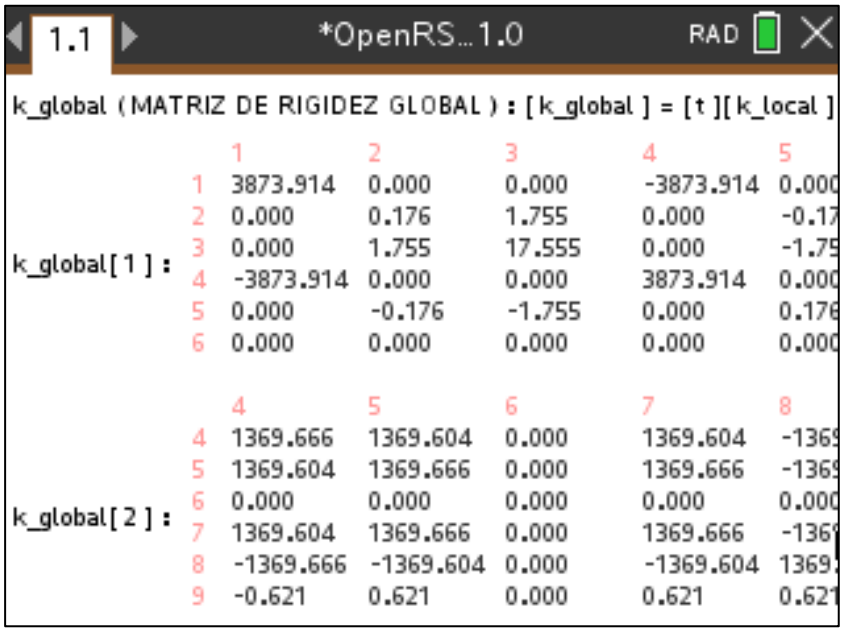
Ventana de matrices de transformación



Nota. Presenta la matriz de transformación de coordenadas locales a globales.

Figura 28

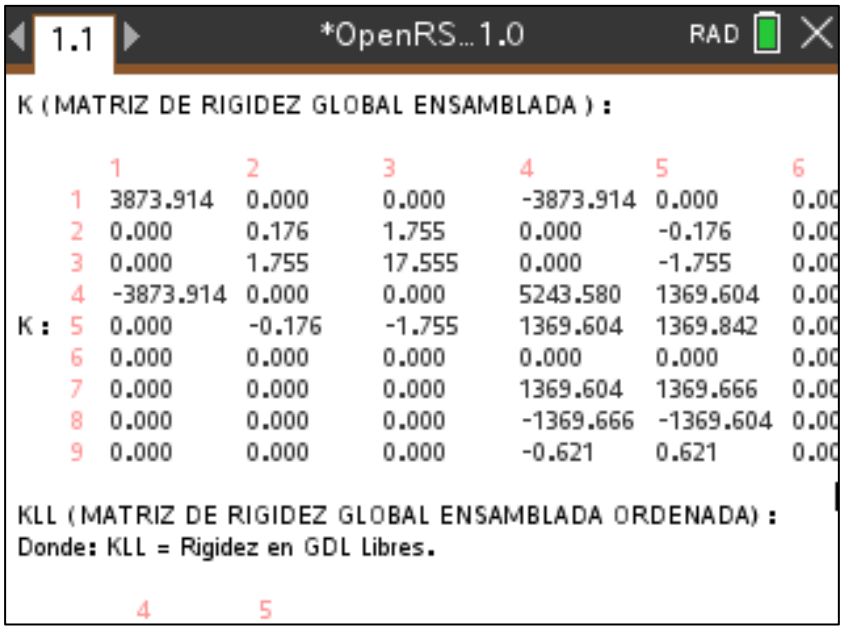
Ventana de matrices de rigidez global



Nota. Muestra las matrices K globales por barra antes del ensamblado.

Figura 29

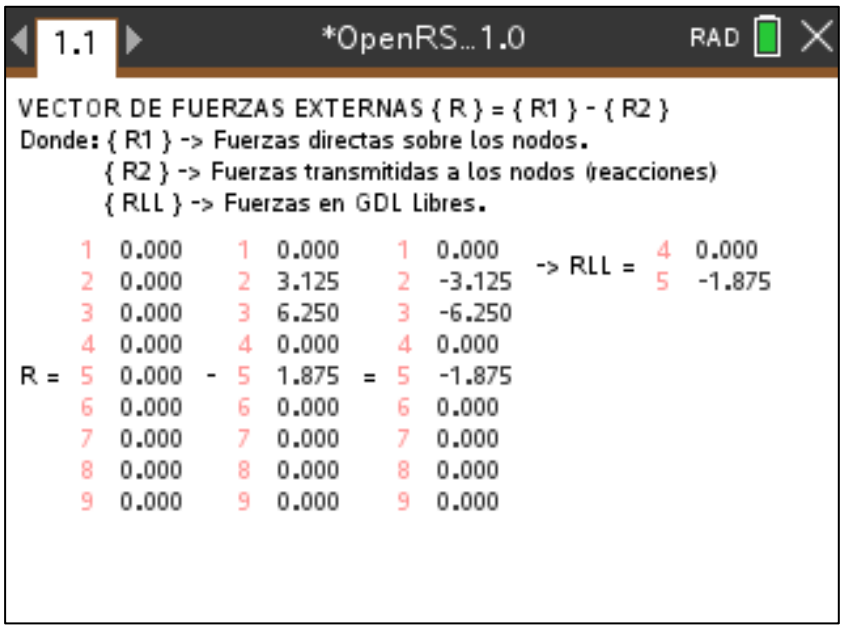
Ventana de matriz de rigidez global ensamblada



Nota. Muestra la matriz de rigidez global ensamblada.

Figura 30

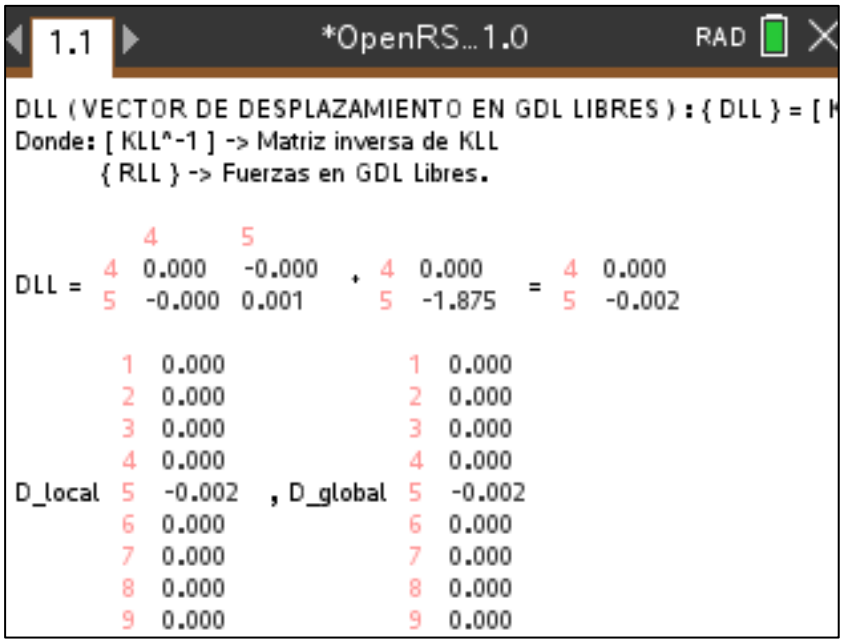
Ventana vectores de fuerzas internas y externas



Nota. Muestra los vectores R1 y R2 que representan cargas nodales y equivalentes.

Figura 31

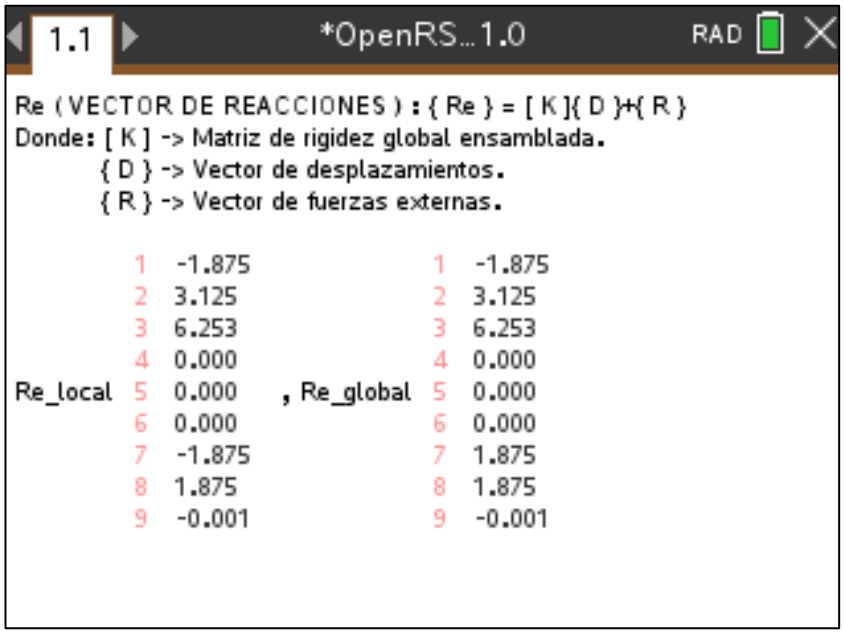
Ventana de vectores de desplazamiento



Nota. Indica los desplazamientos de cada grado de libertad libre.

Figura 32

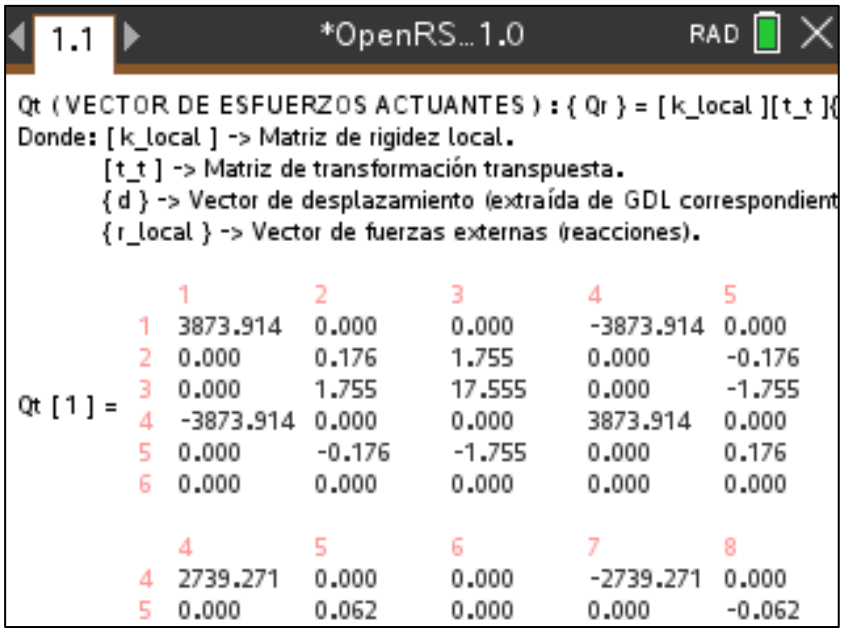
Ventana de vectores de reacciones



Nota. Vector numérico de reacciones estructurales resultantes.

Figura 33

Ventana de vectores de esfuerzos resultantes



Nota. Resultados numéricos de los esfuerzos actuantes en cada barra.

4.1.2. OPTIMIZACIÓN DEL ANÁLISIS Y LÓGICA DE CÁLCULOS

El módulo de análisis estructural de OPENRSE fue desarrollado bajo un enfoque que busca garantizar una ejecución eficiente, precisa y accesible, sin comprometer la versatilidad del software frente a diferentes tipos de estructuras bidimensionales. Una de las limitaciones más comunes en programas comerciales es la necesidad de predefinir el tipo de estructura (marco, armadura, viga continua, etc.), lo cual encierra al usuario en modelos estáticos. En contraposición, OPENRSE implementa una lógica de análisis que permite representar cualquier configuración estructural posible a partir de una sola plataforma funcional, sin necesidad de cambiar de entorno o de adaptar complejamente el modelo.

Este diseño optimizado se fundamenta en dos pilares clave:

- Un sistema dinámico de generación de matrices de rigidez, que se ajusta automáticamente según las condiciones de frontera asignadas a cada barra del modelo.
- Un sistema automatizado de clasificación de grados de libertad, que interpreta con precisión las condiciones de apoyo y determina el comportamiento estructural de cada nodo.
- Un sistema de evaluación precisa de los puntos críticos de esfuerzo, que utiliza funciones avanzadas del entorno TI-Nspire CX CAS para identificar los valores máximos y mínimos de cortante y momento flector, superando las limitaciones de discretización de los programas comerciales.

Sistema de rigidez adaptable por barra

En estructuras hiperestáticas bidimensionales, el comportamiento de cada barra depende de cómo están conectados sus extremos. Para representar con precisión este comportamiento, OpenRSE implementa cuatro funciones distintas que generan la matriz de rigidez local de cada elemento, dependiendo de si los extremos son rígidos o articulados.

El siguiente bloque de código muestra cómo el sistema elige automáticamente la matriz apropiada para cada barra, dentro del método calcular_k_local():

Figura 34

Método para la determinación de condición de frontera

```
function analizar:calcular_k_local()
    local barra = self.datos.barra
    for i, k in ipairs(barra.conexion) do
        local A,I = barra.propiedad[i][1],barra.propiedad[i][2]
        local E,L = barra.propiedad[i][3],barra.longitud[i][1]
        local ci,cj = barra.contorno[i][1],barra.contorno[i][2]
        local k_local

        if not ci and not cj then
            k_local = self:matriz_rigidez_na_na(E,A,I,L)  -- Rígido - Rígido
        elseif ci and not cj then
            k_local = self:matriz_rigidez_a_na(E,A,I,L)   -- Articulado - Rígido
        elseif not ci and cj then
            k_local = self:matriz_rigidez_na_a(E,A,I,L)   -- Rígido - Articulado
        elseif ci and cj then
            k_local = self:matriz_rigidez_a_a(E,A,L)      -- Articulado - Articulado
        end

        table.insert(self.k_local, k_local)
    end
end
```

Nota. Cada una de las funciones llamadas implementa una matriz de rigidez (6x6) diferente, según el comportamiento esperado del elemento estructural.

Figura 35

Método para el caso Rígido – Rígido

```
function analizar:matriz_rigidez_na_na(E,A,I,L)
    local K = {
        {E*A/L,0,0,-E*A/L,0,0},
        {0,12*E*I/L^3,6*E*I/L^2,0,-12*E*I/L^3,6*E*I/L^2},
        {0,6*E*I/L^2,4*E*I/L,0,-6*E*I/L^2,2*E*I/L},
        {-E*A/L,0,0,E*A/L,0,0},
        {0,-12*E*I/L^3,-6*E*I/L^2,0,12*E*I/L^3,-6*E*I/L^2},
        {0,6*E*I/L^2,2*E*I/L,0,-6*E*I/L^2,4*E*I/L}
    }
    return K
end
```

Nota. Matriz completa que incluye rigidez axial, cortante y flexional para ambos extremos empotrados.

Figura 36

Método para el caso Articulado – Rígido

```
function analizar:matriz_rigidez_a_na(E,A,I,L)
    local K = {
        {E*A/L,0,0,-E*A/L,0,0},
        {0,3*E*I/L^3,0,0,-3*E*I/L^3,3*E*I/L^2},
        {0,0,0,0,0,0},
        {-E*A/L,0,0,E*A/L,0,0},
        {0,-3*E*I/L^3,0,0,3*E*I/L^3,-3*E*I/L^2},
        {0,3*E*I/L^2,0,0,-3*E*I/L^2,3*E*I/L}
    }
    return K
end
```

Nota. Un extremo articulado elimina la rotación y reduce la rigidez flexional.

Figura 37

Método para el caso Rígido – Articulado

```
function analizar:matriz_rigidez_na_a(E,A,I,L)
    local K = {
        {E*A/L,0,0,-E*A/L,0,0},
        {0,3*E*I/L^3,3*E*I/L^2,0,-3*E*I/L^3,0},
        {0,3*E*I/L^2,3*E*I/L,0,-3*E*I/L^2,0},
        {-E*A/L,0,0,E*A/L,0,0},
        {0,-3*E*I/L^3,-3*E*I/L^2,0,3*E*I/L^3,0},
        {0,0,0,0,0,0}
    }
    return K
end
```

Nota. Variante del caso anterior con articulación en el nodo opuesto.

Figura 38

Método para el caso Articulado – Articulado

```
function analizar:matriz_rigidez_a_a(E,A,L)
    local K = {
        {E*A/L,0,0,-E*A/L,0,0},
        {0,0,0,0,0,0},
        {0,0,0,0,0,0},
        {-E*A/L,0,0,E*A/L,0,0},
        {0,0,0,0,0,0},
        {0,0,0,0,0,0}
    }
    return K
end
```

Nota. Solo considera esfuerzos axiales. Propio de estructuras tipo armadura.

Estas funciones son llamadas dinámicamente durante el análisis, lo cual permite construir una matriz global de rigidez adaptable, que refleja con precisión el comportamiento individual de cada barra,

considerando su geometría, propiedades mecánicas y condiciones de contorno.

Este sistema optimizado no solo garantiza eficiencia computacional, sino que también permite que OpenRSE sea capaz de analizar cualquier configuración estructural bidimensional, sin imponer limitaciones estructurales ni exigir al usuario conocimientos avanzados de codificación o teoría.

Definición optimizada de apoyos y clasificación automatizada de grados de libertad

Una de las características que optimiza el análisis estructural en OPENRSE es la automatización de la asignación de grados de libertad (GDL) a partir de los tipos de apoyo definidos por el usuario. Este procedimiento reemplaza configuraciones complejas presentes en otros programas, donde se requiere configurar manualmente las restricciones mediante supresión directa de ecuaciones, matrices o condiciones frontera avanzadas. En OPENRSE, este proceso es interpretado automáticamente por el método clasificar_gdl() a partir de la selección numérica de tipo de apoyo desde la interfaz.

El primer paso del proceso consiste en definir una tabla interna que relaciona cada tipo de apoyo con sus respectivas condiciones de restricción en los tres grados de libertad. Este arreglo permite representar apoyos comunes como empotrados, simples, rodillos o condiciones parcialmente restringidas, de forma clara y sistemática.

Figura 39

Arreglo para la clasificación de grados de libertad

```
restricciones = {
    {true,  true,  true},  -- 1: Libre
    {false, false, false}, -- 2: Empotrado
    {false, false, true},  -- 3: Simple
    {true,  false, true},  -- 4: Rodillo en X
    {false, true,  true},  -- 5: Rodillo en Y
    {true,  true,  false}, -- 6: Giro restringido
    {true,  false, false}, -- 7: X libre
    {false, true,  false}  -- 8: Y libre
}
```

Nota. El arreglo restricciones define las posibles combinaciones de libertad y restricción en los grados de libertad de cada nodo (traslación en X, traslación en Y y rotación), permitiendo representar ocho tipos distintos de apoyos estructurales en el análisis bidimensional de estructuras hiperestáticas.

Antes de aplicar las restricciones por tipo de apoyo, el sistema analiza si el nodo está conectado únicamente mediante barras articuladas. En ese caso, su rotación se restringe automáticamente, ya que no se transmite momento en dicho nodo. Este paso evita errores de interpretación estructural sin que el usuario tenga que intervenir.

Figura 40

Bloque de detección automática de nodos articulados

```
for i, _ in ipairs(nodo.coordenada) do
    local cumpleCondicion = true
    for j, conexion in ipairs(barra.conexion) do
        local ni, nj = conexion[1], conexion[2]
        local ci, cj = barra.contorno[j][1], barra.contorno[j][2]
        local articulado = (ni == i and ci) or (nj == i and cj)
        if articulado == false then
            cumpleCondicion = false
        end
    end
    if cumpleCondicion then
        local gdl = ((i - 1) * 3) + 3
        table.insert(self.gdl_articulado, gdl)
    end
end
```

Nota. Este bloque evalúa si un nodo está conectado exclusivamente mediante articulaciones, en cuyo caso se restringe automáticamente su grado de libertad rotacional.

Finalmente, el sistema recorre cada nodo y, según el tipo de apoyo asignado, determina cuáles grados de libertad estarán activos (libres) y cuáles deben restringirse para el análisis. Esta clasificación se realiza integrando tanto la tabla restricciones como la lista de nodos articulados detectados previamente.

Figura 41

Bloque de asignación de GDL libres y restringidos

```
for i, restriccion in ipairs(nodo.restriccion) do
    local tipo = restriccion[1]
    local baseGDL = (i - 1) * 3
    local cond = restricciones[tipo]

    for gdl = 1, 3 do
        local actual = baseGDL + gdl
        local esArticulado = false

        for _, gdlA in ipairs(self.gdl_articulado) do
            if gdlA == actual then esArticulado = true end
        end

        if esArticulado or not cond[gdl] then
            table.insert(self.gdl_restringido, actual)
        else
            table.insert(self.gdl_libre, actual)
        end
    end
end
```

Nota. En este bloque se aplica la lógica de restricciones a cada nodo, generando las listas de GDL libres y restringidos necesarias para el armado de matrices y resolución del sistema estructural.

Esta segmentación del método clasificar_gdl() permite una comprensión progresiva del proceso lógico y evidencia cómo OPENRSE automatiza decisiones estructurales que en otros entornos requieren conocimientos técnicos avanzados. Esta capacidad de interpretar condiciones fronteras complejas con mínima intervención del usuario aumenta las posibilidades de modelado, reduce errores de entrada y mejora la eficiencia operativa general del software.

Evaluación precisa de puntos críticos de cortante y momento

Otro aspecto que requirió optimización en el desarrollo de OPENRSE fue la determinación de los valores máximos y mínimos de

los diagramas de esfuerzos internos, particularmente para cortantes y momentos flectores. En los programas especializados, esta evaluación suele estar limitada a un número reducido de puntos predefinidos por el sistema, lo cual puede dificultar la detección exacta de los valores críticos. Incluso cuando es posible explorar el diagrama mediante herramientas visuales, como barras deslizantes o cursores gráficos, esto no permite al usuario establecer con precisión un punto decimal específico en el que evaluar la ecuación de esfuerzo.

La razón de esta limitación es que muchos programas priorizan velocidad y simplicidad gráfica, evaluando internamente las ecuaciones de esfuerzo solo en ciertos nodos o divisiones equidistantes, en lugar de resolver analíticamente la función. Como resultado, los valores reportados como máximos o mínimos pueden contener un margen de error, especialmente en estructuras con cargas distribuidas no uniformes.

En OPENRSE, esta situación fue resuelta mediante la implementación de un sistema que evalúa de forma precisa los puntos extremos de las ecuaciones de cortante y momento, haciendo uso de funciones avanzadas del entorno de programación TI-Nspire CX CAS. En particular, se utilizan las funciones nfMax y nfMin, propias del lenguaje TI-Basic, que realizan una evaluación numérica optimizada del valor máximo y mínimo de una expresión en un intervalo definido, con precisión de punto flotante y sin requerir un número de subdivisiones preestablecido.

Esta herramienta interna permite al programa localizar los extremos reales de los esfuerzos con alta precisión, sin depender de discretización fija o de exploraciones manuales por parte del usuario.

A continuación, se presenta el código correspondiente:

Figura 42

Método de análisis de cortante máximo y mínimo

```
function analizar:calcular_puntos_criticos_cortante()
    local barra = self.datos.barra
    for i,k in ipairs(barra.conexion) do
        local wa = barra.carga[i][1]
        local wb = barra.carga[i][2]
        local L  = barra.longitud[i][1]
        local cv = self.Qt[i][2][1]
        --
        local a = (-wa+wb)/(2*L)
        local b = wa
        local c = cv
        local a = string.format("%.10f", a)
        local b = string.format("%.10f", b)
        local c = string.format("%.10f", c)
        --
        math.eval("xmax:=nfMax("..a.."*x^2+"..b.."*x+"..c..",x,0,"..L..")")
        math.eval("xmin:=nfMin("..a.."*x^2+"..b.."*x+"..c..",x,0,"..L..")")
        local xmax = tonumber(math.eval("xmax"))
        local xmin = tonumber(math.eval("xmin"))
        table.insert(self.puntos_criticos_cortante,{xmax,xmin})
        math.eval("DelVar xmax,xmin")
    end
end
```

Nota. El método evalúa los puntos extremos del diagrama de esfuerzo cortante mediante los métodos nfMax y nfMin, exclusivos del sistema TI-Nspire CX CAS.

Figura 43

Método de análisis flector máximo y mínimo

```
function analizar:calcular_puntos_criticos_momento()
    local barra = self.datos.barra
    for i,k in ipairs(barra.conexion) do
        local wa = barra.carga[i][1]
        local wb = barra.carga[i][2]
        local L  = barra.longitud[i][1]
        local cv = self.Qt[i][2][1]
        local cm = self.Qt[i][3][1]
        --
        local a = -(-wa+wb)/(6*L)
        local b = -wa/2
        local c = -cv
        local d = cm
        local a = string.format("%.10f", a)
        local b = string.format("%.10f", b)
        local c = string.format("%.10f", c)
        local d = string.format("%.10f", d)
        --
        math.eval("xmax:=nfMax("..tostring(a).."*x^3+"..tostring(b).."*x^2+"..tostring(c).."*x+"..
        math.eval("xmin:=nfMin("..tostring(a).."*x^3+"..tostring(b).."*x^2+"..tostring(c).."*x+"..
        local xmax = tonumber(math.eval("xmax"))
        local xmin = tonumber(math.eval("xmin"))
        table.insert(self.puntos_criticos_momento,{xmax,xmin})
        math.eval("DelVar xmax,xmin")
    end
end
```

Nota. Este método permite obtener los valores extremos del momento flector para cada barra, con una precisión superior al realizarse mediante cálculo simbólico optimizado en el entorno TI-Nspire.

Este sistema de evaluación automatizada, sin necesidad de definir un número fijo de puntos o resoluciones gráficas, brinda mayor precisión

que los métodos implementados en software especializado, donde los valores extremos se aproximan mediante discretización o interpolación visual. En las pruebas realizadas, los valores obtenidos por OPENRSE han sido verificados manualmente y presentan una precisión superior, permitiendo análisis más confiables, especialmente en el diseño de elementos sometidos a máximos esfuerzos internos.

4.1.3. PROCESAMIENTO Y ANÁLISIS COMPARATIVO DE RESULTADOS ESTRUCTURALES

Con el objetivo de analizar y comparar los resultados estructurales obtenidos con OpenRSE frente a los principales softwares comerciales de referencia (SAP2000 y Robot Structural), se desarrollaron diez modelos representativos de diversas condiciones estructurales reales. La selección de estos modelos fue dirigida estratégicamente para abarcar configuraciones variadas en términos de geometría, condiciones de contorno, materiales y sistemas estructurales, con el fin de cubrir un amplio espectro de escenarios posibles en el análisis bidimensional de estructuras hiperestáticas. Este enfoque permite no solo procesar y contrastar los resultados obtenidos, sino también evaluar con solidez el desempeño y la precisión del software desarrollado frente a herramientas reconocidas del mercado.

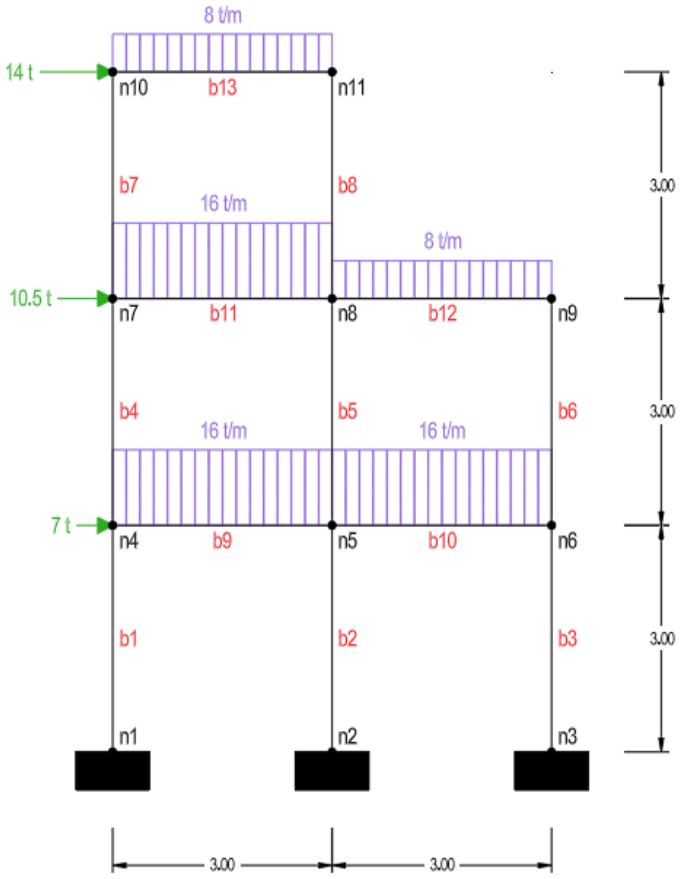
Modelo 01: Edificación de tres niveles

Tabla 5

Características y Propiedades del Modelo 01

Modelo: Edificación de tres niveles	Propiedades físicas:
Tipo: pórtico	M. de elasticidad: 2173706.5 t/m²
Numero de barras: 9	Sección: 0.4 x 0.4 m
Numero de nodos: 13	Área: 0.160 m²
Observaciones: Se usa el sistema internaciones de unidades (SI)	Momento de inercia: 0.002133 m⁴ (Concreto f'c 210 kg/cm2)

Representación del Modelo Matemático



Nota. El modelo 01 representa una edificación tipo pórtico de tres niveles con carga distribuida en vigas. Se especifican sus propiedades físicas y estructurales, así como su representación matemática con numeración de nodos y elementos.

A continuación, se presentan los resultados individuales obtenidos en los tres programas de análisis estructural utilizados: SAP2000, Robot

Structural y OpenRSE. Para facilitar la revisión, se han seleccionado los nodos y barras más representativos del modelo, permitiendo una lectura concisa y relevante de los parámetros clave como desplazamientos, reacciones y esfuerzos internos.

Tabla 6

Resultados del análisis estructural – Modelo 01 (SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	7.74	28.92	16.52	1	28.916	7.741	16.518
2	0.000	0.000	0.000	12.40	107.37	21.04	3	55.718	11.360	20.031
3	0.000	0.000	0.000	11.36	55.72	20.03	5	57.924	11.333	17.330
4	0.871	0.025	0.003	0.00	0.00	0.00	8	19.541	10.609	16.072
6	0.870	0.048	0.002	0.00	0.00	0.00	10	0.188	35.220	29.374
8	1.891	0.143	0.001	0.00	0.00	0.00	11	11.896	34.881	25.779
10	2.836	0.044	0.002	0.00	0.00	0.00	-	-	-	-
11	2.827	0.159	0.001	0.00	0.00	0.00	-	-	-	-

Nota. Resultados del análisis del Modelo 01 generados por el software SAP2000.

Tabla 7

Resultados del análisis estructural – Modelo 01 (Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	7.70	28.92	16.26	1	28.920	7.698	16.264
2	0.000	0.000	0.000	12.45	107.22	21.01	3	55.861	11.351	19.905
3	0.000	0.000	0.000	11.35	55.86	19.91	5	57.848	11.384	17.387
4	0.831	0.025	0.003	0.00	0.00	0.00	8	19.514	10.638	16.093
6	0.830	0.048	0.002	0.00	0.00	0.00	10	0.140	35.330	29.544
8	1.805	0.142	0.001	0.00	0.00	0.00	11	11.957	34.864	25.849
10	2.709	0.044	0.002	0.00	0.00	0.00	-	-	-	-
11	2.699	0.159	0.001	0.00	0.00	0.00	-	-	-	-

Nota. Resultados del análisis del Modelo 01 generados por el software Autodesk Robot Structural Analysis.

Tabla 8

Resultados del análisis estructural – Modelo 01 (OpenRSE)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	7.70	28.92	16.26	1	28.920	7.698	16.264
2	0.000	0.000	0.000	12.45	107.22	21.01	3	55.861	11.351	19.905
3	0.000	0.000	0.000	11.35	55.86	19.91	5	57.848	11.384	17.387
4	0.831	0.025	0.003	0.00	0.00	0.00	8	19.514	10.638	16.093
6	0.830	0.048	0.002	0.00	0.00	0.00	10	0.140	35.330	29.544
8	1.805	0.142	0.001	0.00	0.00	0.00	11	11.957	34.865	25.850
10	2.709	0.044	0.002	0.00	0.00	0.00	-	-	-	-
11	2.700	0.159	0.001	0.00	0.00	0.00	-	-	-	-

Nota. Resultados del análisis del Modelo 01 generados por el software OpenRSE, desarrollado en la presente investigación.

Para validar los resultados de OpenRSE, se presentan las diferencias absolutas, ya que permiten comparaciones directas sin distorsión en magnitudes pequeñas o grandes.

Tabla 9

Errores absolutos – Modelo 01 (OpenRSE vs. SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.042	0.000	0.256	1	0.004	0.043	0.254
2	0.000	0.000	0.000	0.051	0.150	0.034	3	0.143	0.009	0.126
3	0.000	0.000	0.000	0.009	0.141	0.125	5	0.076	0.051	0.057
4	0.040	0.000	0.000	0.000	0.000	0.000	8	0.027	0.029	0.021
6	0.040	0.000	0.000	0.000	0.000	0.000	10	0.048	0.110	0.170
8	0.086	0.001	0.000	0.000	0.000	0.000	11	0.061	0.016	0.071
10	0.127	0.000	0.000	0.000	0.000	0.000	-	-	-	-
11	0.127	0.000	0.000	0.000	0.000	0.000	-	-	-	-

Nota. Errores absolutos calculados, basados en la comparación de los resultados de OpenRSE.

Tabla 10

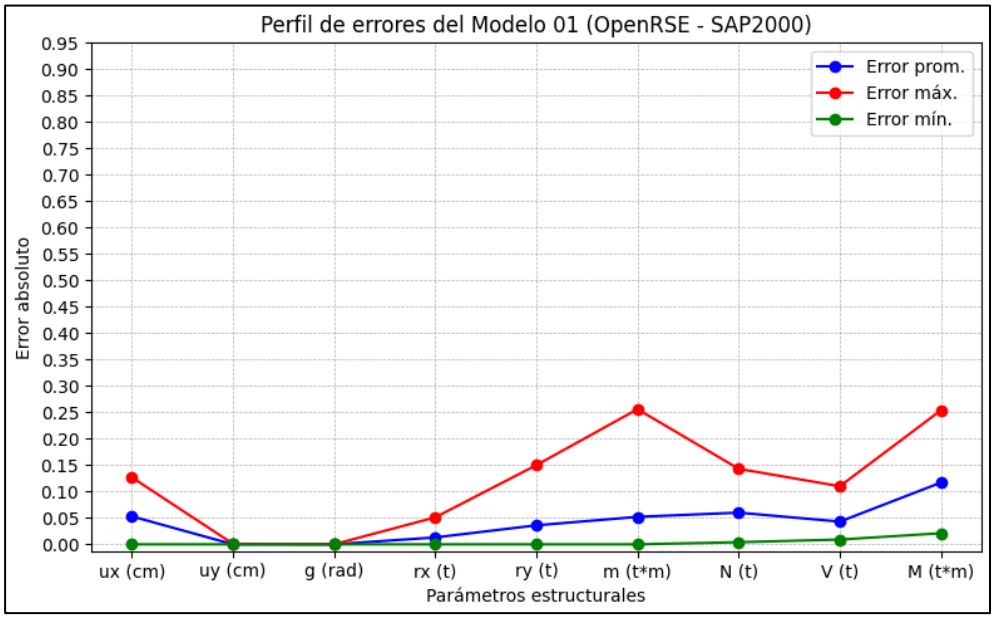
Resumen estadístico de errores absolutos – Modelo 01 (OpenRSE vs. SAP2000)

Medida	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.053	0.000	0.000	0.013	0.036	0.052	0.060	0.043	0.117
Err. max.	0.127	0.001	0.000	0.051	0.150	0.256	0.143	0.110	0.254
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	0.004	0.009	0.021

Nota. El mayor error registrado corresponde al momento flector (0.256 t*m), seguido por la normal (0.143 t), ambos dentro de márgenes aceptables para análisis estructurales, sin comprometer la precisión del modelo.

Figura 44

Perfil de errores absolutos – Modelo 01 (OpenRSE vs. SAP2000)



Nota. El gráfico evidencia que los errores máximos se concentran principalmente en los momentos flectores y la normal, mientras que en la mayoría de los parámetros los errores mínimos son nulos y los errores promedio se mantienen bajos.

Tabla 11

Errores absolutos – Modelo 01 (OpenRSE vs. Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.000	0.000	0.000	1	0.000	0.000	0.000
2	0.000	0.000	0.000	0.000	0.001	0.000	3	0.000	0.000	0.000
3	0.000	0.000	0.000	0.000	0.000	0.000	5	0.000	0.000	0.000
4	0.000	0.000	0.000	0.000	0.000	0.000	8	0.000	0.000	0.000
6	0.000	0.000	0.000	0.000	0.000	0.000	10	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	11	0.000	0.001	0.001
10	0.000	0.000	0.000	0.000	0.000	0.000	-	-	-	-
11	0.001	0.000	0.000	0.000	0.000	0.000	-	-	-	-

Nota. Errores absolutos calculado, basados en la comparación de los resultados de OpenRSE

Tabla 12

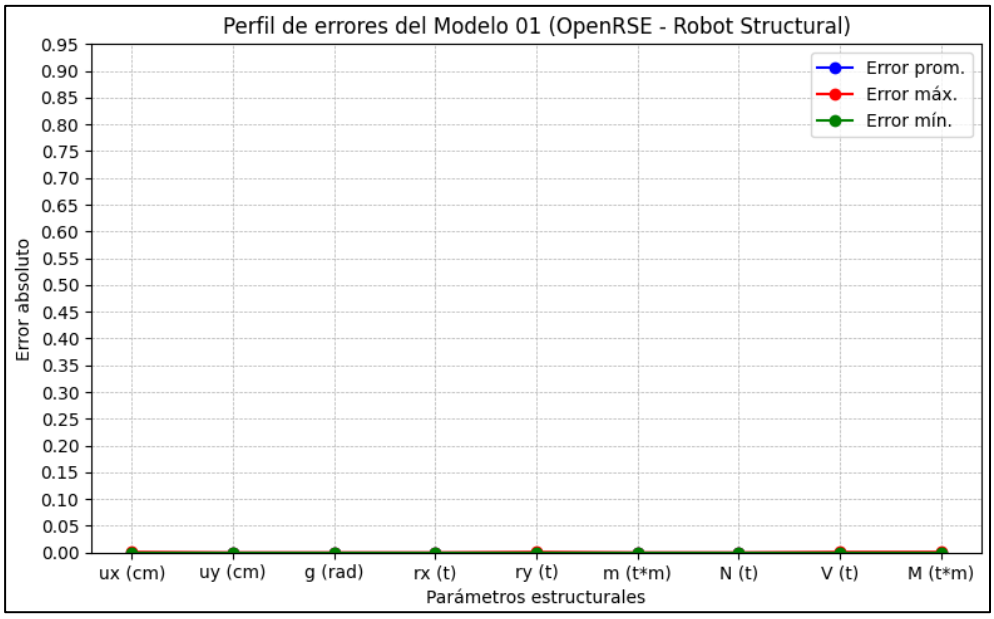
Resumen estadístico de errores absolutos – Modelo 01 (OpenRSE vs. Robot Structural)

Medida	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Err. max.	0.001	0.000	0.000	0.000	0.001	0.000	0.000	0.001	0.001
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Nota. Los resultados evidencian una coincidencia prácticamente exacta, con errores absolutos nulos o despreciables en todos los parámetros estructurales evaluados.

Figura 45

Perfil de errores absolutos – Modelo 01 (OpenRSE vs. Robot Structural)



Nota. Todos los errores resultaron cercanos a cero, lo que refleja una alta precisión en los resultados obtenidos por OpenRSE.

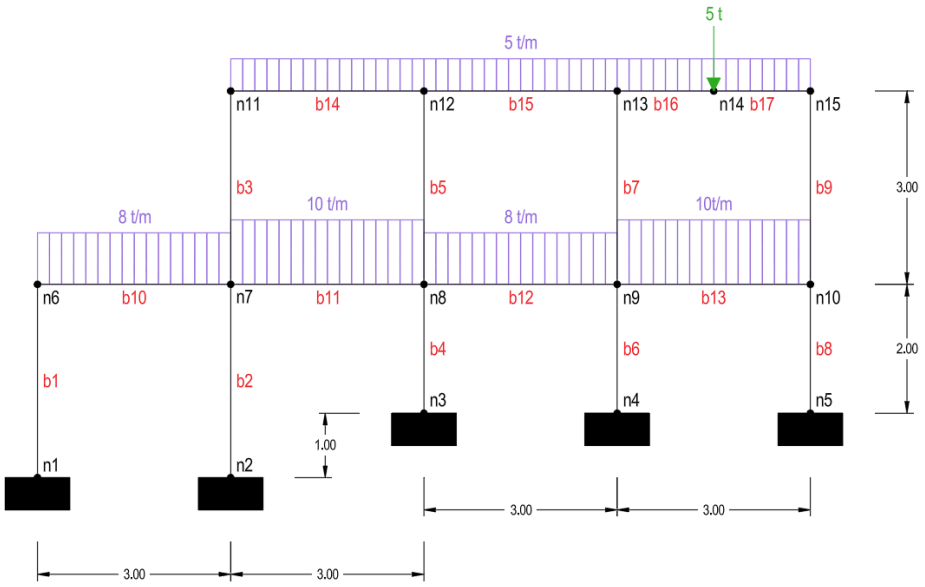
Modelo 02: Edificación con desnivel

Tabla 13

Características y Propiedades del Modelo 02

Modelo: Edificación con desnivel	Propiedades físicas:
Tipo: Pórtico	M. de elasticidad: 2173706.5 t/m²
Numero de barras: 17	Sección: 0.4 x 0.4 m
Numero de nodos: 15	Área: 0.160 m²
Observaciones: Se usa el sistema internaciones de unidades (SI)	Momento de inercia: 0.002133 m⁴ (Concreto f'c 210 kg/cm2)

Representación del Modelo Matemático



Nota. El modelo 02 representa una edificación tipo pórtico con variación de alturas entre columnas, simulando un comportamiento estructural con desnivel. La distribución de cargas y geometría permite analizar la interacción entre niveles y evaluar el efecto de discontinuidades en altura sobre la rigidez global del sistema.

A continuación, se presentan los resultados individuales obtenidos en los tres programas de análisis estructural utilizados: SAP2000, Robot Structural y OpenRSE. Para facilitar la revisión, se han seleccionado los nodos y barras más representativos del modelo, permitiendo una lectura concisa y relevante de los parámetros clave como desplazamientos, reacciones y esfuerzos internos.

Tabla 14

Resultados del análisis estructural – Modelo 02 (SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	1.70	10.90	1.69	1	10.900	1.703	3.418
2	0.000	0.000	0.000	0.04	35.29	0.01	4	42.441	0.252	0.462
3	0.000	0.000	0.000	0.25	42.44	0.04	10	1.703	13.100	6.718
4	0.000	0.000	0.000	0.58	45.86	0.49	12	0.267	12.184	6.479
5	0.000	0.000	0.000	1.99	23.51	1.13	14	1.184	8.085	4.050
6	0.003	0.009	0.001	0.00	0.00	0.00	16	2.128	10.776	5.768
8	0.005	0.024	0.000	0.00	0.00	0.00	-	-	-	-
13	0.030	0.042	0.000	0.00	0.00	0.00	-	-	-	-

Nota. Resultados del análisis del Modelo 02 generados por el software SAP2000.

Tabla 15

Resultados del análisis estructural – Modelo 02 (Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	1.74	10.90	1.77	1	10.904	1.740	3.447
2	0.000	0.000	0.000	0.05	35.30	0.00	4	42.448	0.247	0.448
3	0.000	0.000	0.000	0.25	42.45	0.05	10	1.740	13.096	6.734
4	0.000	0.000	0.000	0.63	45.83	0.54	12	0.298	12.195	6.483
5	0.000	0.000	0.000	2.07	23.52	1.26	14	1.194	8.089	4.068
6	0.003	0.009	0.001	0.00	0.00	0.00	16	2.145	10.772	5.789
8	0.005	0.024	0.000	0.00	0.00	0.00	-	-	-	-
13	0.029	0.042	0.000	0.00	0.00	0.00	-	-	-	-

Nota. Resultados del análisis del Modelo 02 generados por el software Autodesk Robot Structural Analysis.

Tabla 16

Resultados del análisis estructural – Modelo 02 (OpenRSE)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	1.74	10.90	1.77	1	10.904	1.740	3.446
2	0.000	0.000	0.000	0.05	35.30	0.00	4	42.448	0.247	0.448

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
3	0.000	0.000	0.000	0.25	42.45	0.05	10	1.740	13.096	6.734
4	0.000	0.000	0.000	0.63	45.83	0.54	12	0.298	12.195	6.483
5	0.000	0.000	0.000	2.07	23.52	1.26	14	1.194	8.089	4.068
6	0.003	0.009	0.001	0.00	0.00	0.00	16	2.145	10.772	5.789
8	0.005	0.024	0.000	0.00	0.00	0.00	-	-	-	-
13	0.029	0.042	0.000	0.00	0.00	0.00	-	-	-	-

Nota. Resultados del análisis del Modelo 02 generados por el software OpenRSE, desarrollado en la presente investigación.

Para validar los resultados de OpenRSE, se presentan las diferencias absolutas, ya que permiten comparaciones directas sin distorsión en magnitudes pequeñas o grandes.

Tabla 17

Errores absolutos – Modelo 02 (OpenRSE vs. SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.040	0.004	0.082	1	0.004	0.037	0.028
2	0.000	0.000	0.000	0.009	0.005	0.010	4	0.007	0.005	0.014
3	0.000	0.000	0.000	0.003	0.008	0.007	10	0.037	0.004	0.016
4	0.000	0.000	0.000	0.045	0.028	0.051	12	0.031	0.011	0.004
5	0.000	0.000	0.000	0.079	0.012	0.126	14	0.010	0.004	0.018
6	0.000	0.000	0.000	0.000	0.000	0.000	16	0.017	0.004	0.021
8	0.000	0.000	0.000	0.000	0.000	0.000	-	-	-	-
13	0.001	0.000	0.000	0.000	0.000	0.000	-	-	-	-

Nota. Errores absolutos calculados, basados en la comparación de los resultados de OpenRSE.

Tabla 18

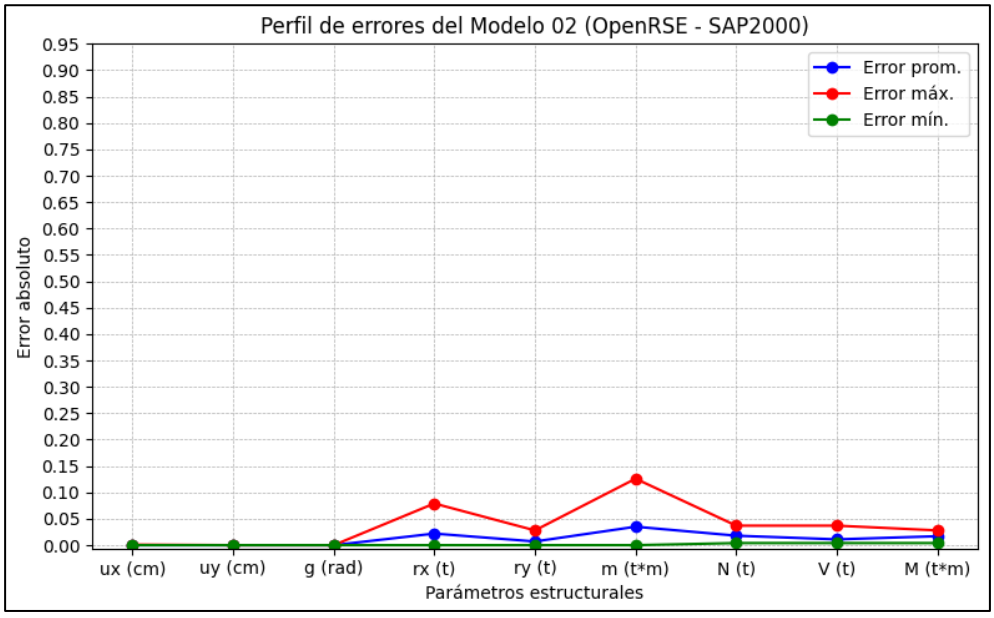
Resumen estadístico de errores absolutos – Modelo 02 (OpenRSE vs. SAP2000)

Medida	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.000	0.000	0.000	0.022	0.007	0.035	0.018	0.011	0.017
Err. max.	0.001	0.000	0.000	0.079	0.028	0.126	0.037	0.037	0.028
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	0.004	0.004	0.004

Nota. El mayor error registrado corresponde al momento flector (0.126 t*m), seguido por la reacción vertical (0.079 t), ambos dentro de márgenes aceptables para análisis estructurales, sin comprometer la precisión del modelo.

Figura 46

Perfil de errores absolutos – Modelo 02 (OpenRSE vs. SAP2000)



Nota. El gráfico evidencia que los errores máximos se concentran principalmente en los momentos flectores y reacciones verticales, mientras que en la mayoría de los parámetros los errores mínimos son nulos y los errores promedio se mantienen bajos.

Tabla 19

Errores absolutos – Modelo 02 (OpenRSE vs. Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.000	0.000	0.000	1	0.000	0.000	0.001
2	0.000	0.000	0.000	0.000	0.000	0.000	4	0.000	0.000	0.000
3	0.000	0.000	0.000	0.000	0.000	0.000	10	0.000	0.000	0.000
4	0.000	0.000	0.000	0.000	0.000	0.000	12	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000	0.000	14	0.000	0.000	0.000
6	0.000	0.000	0.000	0.000	0.000	0.000	16	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	-	-	-	-
13	0.000	0.000	0.000	0.000	0.000	0.000	-	-	-	-

Nota. Errores absolutos calculado, basados en la comparación de los resultados de OpenRSE

Tabla 20

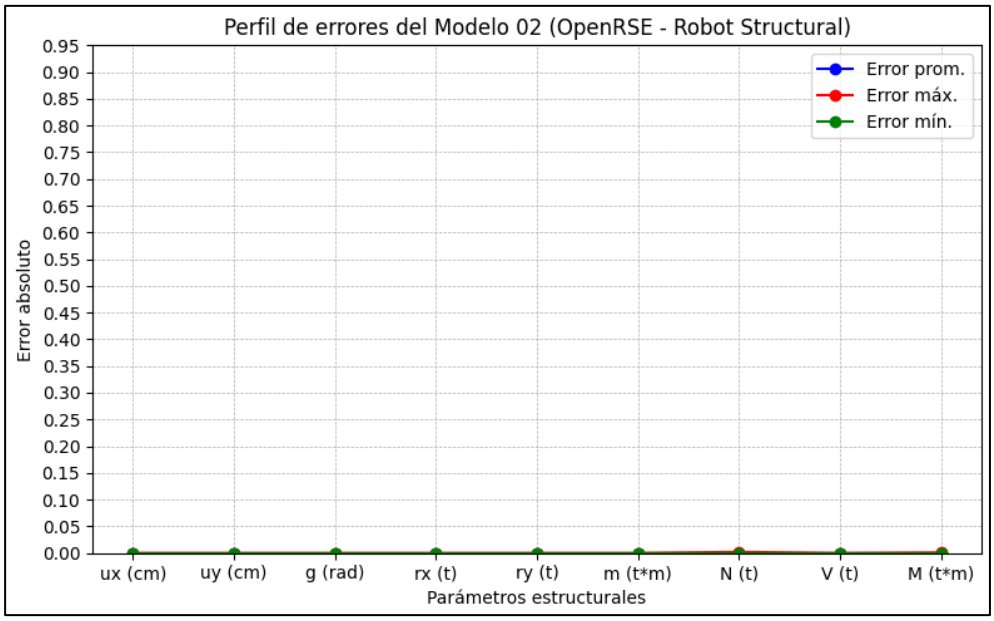
Resumen estadístico de errores absolutos – Modelo 02 (OpenRSE vs. Robot Structural)

Medida	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Err. max.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Nota. Los resultados evidencian una coincidencia prácticamente exacta, con errores absolutos nulos o despreciables en todos los parámetros estructurales evaluados.

Figura 47

Perfil de errores absolutos – Modelo 02 (OpenRSE vs. Robot Structural)



Nota. Todos los errores resultaron cercanos a cero, lo que refleja una alta precisión en los resultados obtenidos por OpenRSE.

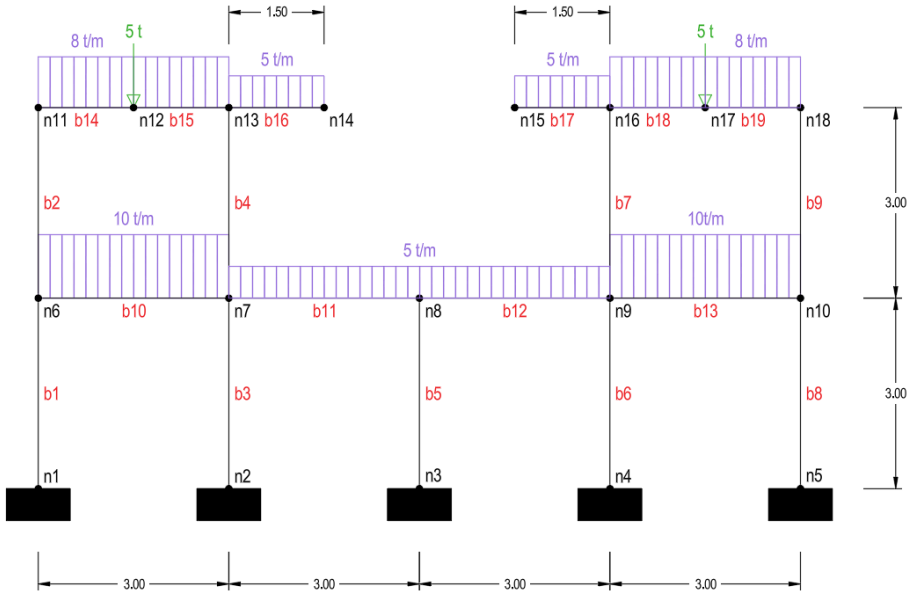
Modelo 03: Edificación de dos niveles con voladizo

Tabla 21

Características y Propiedades del Modelo 03

Modelo: Edificación de dos niveles con voladizo Tipo: Pórtico Numero de barras: 19 Numero de nodos: 18 Observaciones: Se usa el sistema internaciones de unidades (SI)	Propiedades físicas: M. de elasticidad: 2173706.5 t/m² Sección: 0.4 x 0.4 m Área: 0.160 m² Momento de inercia: 0.002133 m⁴ (Concreto f'c 210 kg/cm2)

Representación del Modelo Matemático



Nota. El modelo 03 representa una estructura tipo pórtico de dos niveles con voladizo lateral, lo que permite analizar la distribución de esfuerzos en sistemas extendidos sin apoyo directo. Esta configuración es común en edificaciones con salientes arquitectónicas o balcones estructurales, y resulta útil para evaluar la eficiencia del software ante condiciones de carga asimétricas

A continuación, se presentan los resultados individuales obtenidos en los tres programas de análisis estructural utilizados: SAP2000, Robot Structural y OpenRSE. Para facilitar la revisión, se han seleccionado los nodos y barras más representativos del modelo, permitiendo una lectura concisa y relevante de los parámetros clave como desplazamientos, reacciones y esfuerzos internos.

Tabla 22

Resultados del análisis estructural – Modelo 03 (SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	1.38	27.11	1.34	1	27.111	1.375	2.786
2	0.000	0.000	0.000	0.35	46.69	0.35	8	27.111	1.375	2.786
3	0.000	0.000	0.000	0.00	15.39	0.00	10	0.842	15.714	7.673
4	0.000	0.000	0.000	0.35	46.69	0.35	12	1.025	7.697	4.220
5	0.000	0.000	0.000	1.38	27.11	1.34	16	0.000	7.500	5.625
8	0.000	0.013	0.000	0.00	0.00	0.00	19	2.216	12.825	6.333
14	0.082	0.119	0.001	0.00	0.00	0.00	-	-	-	-
18	0.084	0.034	0.001	0.00	0.00	0.00	-	-	-	-

Nota. Resultados del análisis del Modelo 03 generados por el software SAP2000.

Tabla 23

Resultados del análisis estructural – Modelo 03 (Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	1.39	27.12	1.39	1	27.117	1.386	2.774
2	0.000	0.000	0.000	0.36	46.67	0.37	8	27.117	1.386	2.774
3	0.000	0.000	0.000	0.00	15.43	0.00	10	0.863	15.703	7.691
4	0.000	0.000	0.000	0.36	46.67	0.37	12	1.029	7.717	4.244
5	0.000	0.000	0.000	1.39	27.12	1.39	16	0.000	7.500	5.625
8	0.000	0.013	0.000	0.00	0.00	0.00	19	2.250	12.820	6.288
14	0.080	0.114	0.001	0.00	0.00	0.00	-	-	-	-
18	0.081	0.034	0.001	0.00	0.00	0.00	-	-	-	-

Nota. Resultados del análisis del Modelo 03 generados por el software Autodesk Robot Structural Analysis.

Tabla 24

Resultados del análisis estructural – Modelo 03 (OpenRSE)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	1.39	27.12	1.39	1	27.117	1.386	2.774
2	0.000	0.000	0.000	0.36	46.67	0.37	8	27.117	1.386	2.774
3	0.000	0.000	0.000	0.00	15.43	0.00	10	0.863	15.703	7.691
4	0.000	0.000	0.000	0.36	46.67	0.37	12	1.029	7.717	4.244
5	0.000	0.000	0.000	1.39	27.12	1.39	16	0.000	7.500	5.625
8	0.000	0.013	0.000	0.00	0.00	0.00	19	2.250	12.820	6.288
14	0.079	0.114	0.001	0.00	0.00	0.00	-	-	-	-
18	0.081	0.034	0.001	0.00	0.00	0.00	-	-	-	-

Nota. Resultados del análisis del Modelo 03 generados por el software OpenRSE, desarrollado en la presente investigación.

Para validar los resultados de OpenRSE, se presentan las diferencias absolutas, ya que permiten comparaciones directas sin distorsión en magnitudes pequeñas o grandes.

Tabla 25

Errores absolutos – Modelo 03 (OpenRSE vs. SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.011	0.006	0.047	1	0.006	0.011	0.012
2	0.000	0.000	0.000	0.008	0.025	0.016	8	0.006	0.011	0.012
3	0.000	0.000	0.000	0.000	0.039	0.000	10	0.021	0.011	0.018
4	0.000	0.000	0.000	0.008	0.025	0.016	12	0.004	0.020	0.024
5	0.000	0.000	0.000	0.011	0.006	0.047	16	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	19	0.034	0.005	0.045
14	0.003	0.005	0.000	0.000	0.000	0.000	-	-	-	-
18	0.003	0.000	0.000	0.000	0.000	0.000	-	-	-	-

Nota. Errores absolutos calculados, basados en la comparación de los resultados de OpenRSE.

Tabla 26

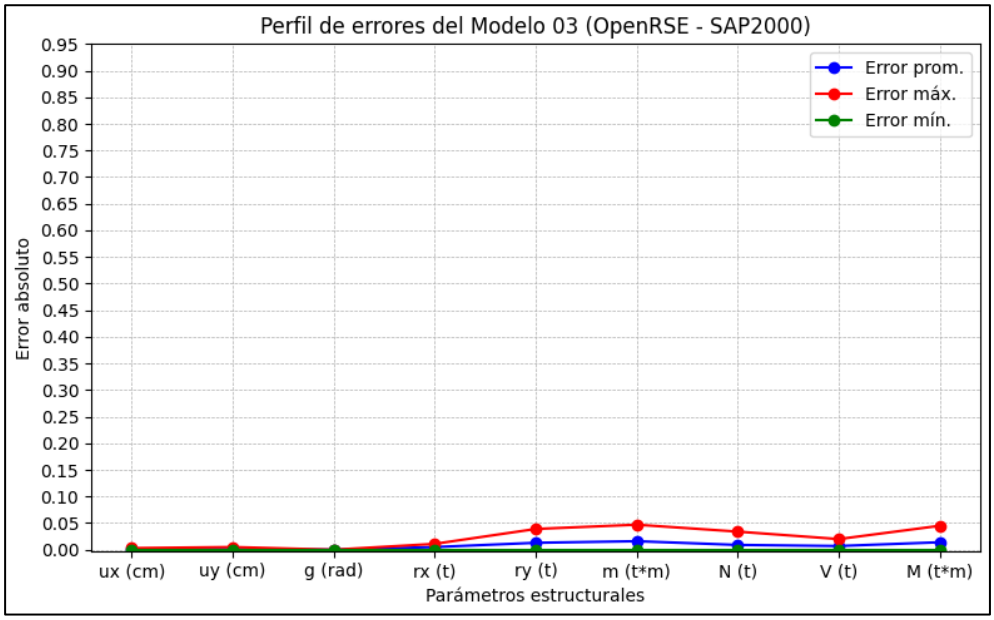
Resumen estadístico de errores absolutos – Modelo 03 (OpenRSE vs. SAP2000)

Medida	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.001	0.001	0.000	0.005	0.013	0.016	0.012	0.010	0.019
Err. max.	0.003	0.005	0.000	0.011	0.039	0.047	0.034	0.020	0.045
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Nota. El mayor error registrado corresponde al momento flector (0.047 t*m), dentro de márgenes aceptables para análisis estructurales, sin comprometer la precisión del modelo.

Figura 48

Perfil de errores absolutos – Modelo 03 (OpenRSE vs. SAP2000)



Nota. El gráfico evidencia que los errores máximos se concentran principalmente en los momentos flectores, mientras que en la mayoría de los parámetros los errores mínimos son nulos y los errores promedio se mantienen bajos.

Tabla 27

Errores absolutos – Modelo 03 (OpenRSE vs. Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.000	0.000	0.000	1	0.000	0.000	0.000
2	0.000	0.000	0.000	0.000	0.001	0.000	8	0.000	0.000	0.000
3	0.000	0.000	0.000	0.000	0.000	0.000	10	0.000	0.000	0.000
4	0.000	0.000	0.000	0.000	0.001	0.000	12	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000	0.000	16	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	19	0.000	0.000	0.000
14	0.001	0.000	0.000	0.000	0.000	0.000	-	-	-	-
18	0.000	0.000	0.000	0.000	0.000	0.000	-	-	-	-

Nota. Errores absolutos calculado, basados en la comparación de los resultados de OpenRSE.

Tabla 28

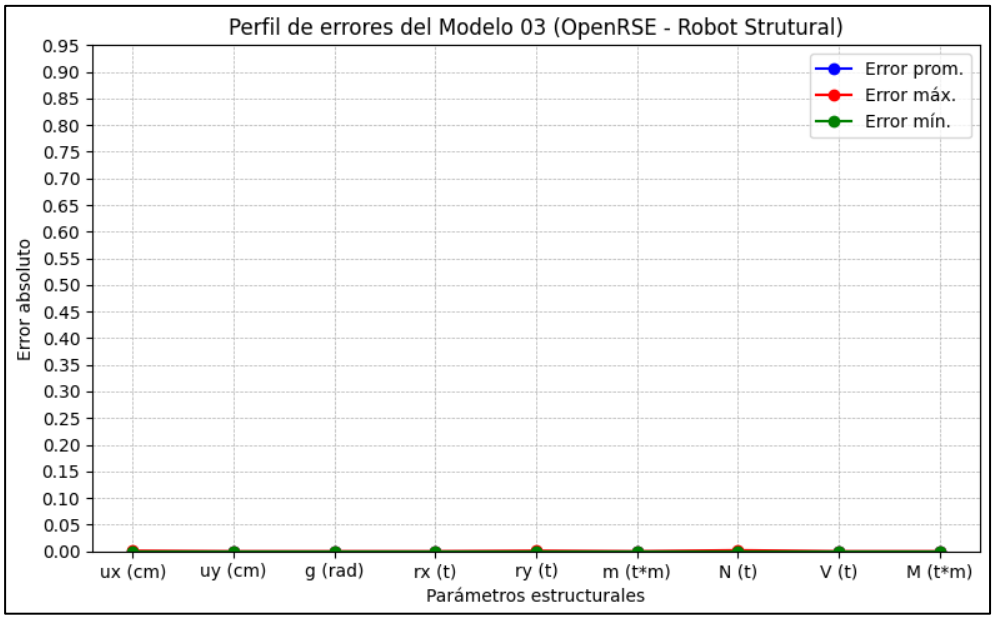
Resumen estadístico de errores absolutos – Modelo 03 (OpenRSE vs. Robot Structural)

Medida	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Err. max.	0.001	0.000	0.000	0.000	0.001	0.000	0.000	0.000	0.000
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Nota. Los resultados evidencian una coincidencia prácticamente exacta, con errores absolutos nulos o despreciables en todos los parámetros estructurales evaluados.

Figura 49

Perfil de errores absolutos – Modelo 03 (OpenRSE vs. Robot Structural)



Nota. Todos los errores resultaron cercanos a cero, lo que refleja una alta precisión en los resultados obtenidos por OpenRSE.

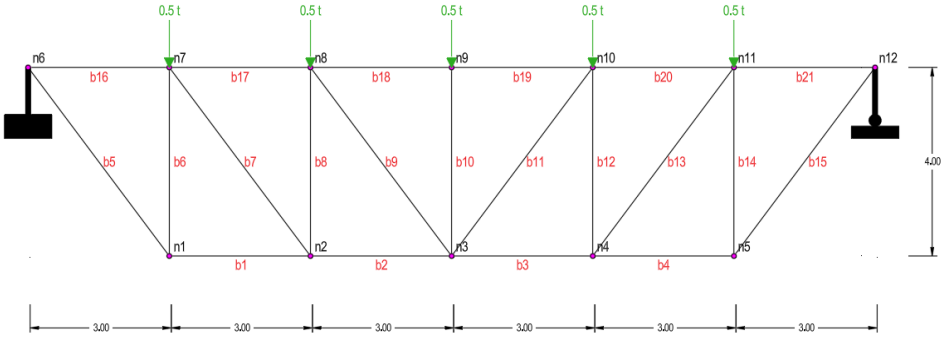
Modelo 04: Plataforma de carga inferior

Tabla 29

Características y Propiedades del Modelo 04

Modelo: Plataforma de carga inferior	Propiedades físicas:
Tipo: Cercha Pratt	M. de elasticidad: 20389019 t/m²
Numero de barras: 21	Sección: 0.1 x 0.1 x 0.005 m
Numero de nodos: 12	Área: 0.0019 m²
Observaciones: Se usa el sistema internaciones de unidades (SI)	Momento de inercia: 0.00000287 m⁴ (Acero A36)

Representación del Modelo Matemático



Nota. El modelo 04 representa una plataforma estructural inferior compuesta por una cercha tipo Pratt, sometida a cargas verticales puntuales distribuidas sobre los nudos superiores. Esta configuración es común en sistemas de soporte de entrepisos, plataformas técnicas o pasarelas suspendidas, donde la estructura principal se encuentra por debajo del plano de carga.

A continuación, se presentan los resultados individuales obtenidos en los tres programas de análisis estructural utilizados: SAP2000, Robot Structural y OpenRSE. Para facilitar la revisión, se han seleccionado los nodos y barras más representativos del modelo, permitiendo una lectura concisa y relevante de los parámetros clave como desplazamientos, reacciones y esfuerzos internos.

Tabla 30

Resultados del análisis estructural – Modelo 04 (SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.051	0.063	0.000	0.00	0.00	0.00	1	0.938	0.000	0.000
3	0.032	0.141	0.000	0.00	0.00	0.00	3	1.500	0.000	0.000
6	0.000	0.000	0.000	0.00	1.25	0.00	16	0.938	0.000	0.000
8	0.019	0.126	0.000	0.00	0.00	0.00	19	1.688	0.000	0.000
11	0.057	0.076	0.000	0.00	0.00	0.00	-	-	-	-
12	0.064	0.000	0.000	0.00	1.25	0.00	-	-	-	-

Nota. Resultados del análisis del Modelo 04 generados por el software SAP2000.

Tabla 31

Resultados del análisis estructural – Modelo 04 (Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.051	0.063	0.000	0.00	0.00	0.00	1	0.938	0.000	0.000
3	0.032	0.141	0.000	0.00	0.00	0.00	3	1.500	0.000	0.000
6	0.000	0.000	0.000	0.00	1.25	0.00	16	0.937	0.000	0.000
8	0.019	0.126	0.000	0.00	0.00	0.00	19	1.687	0.000	0.000
11	0.057	0.076	0.000	0.00	0.00	0.00	-	-	-	-
12	0.064	0.000	0.000	0.00	1.25	0.00	-	-	-	-

Nota. Resultados del análisis del Modelo 04 generados por el software Autodesk Robot Structural Analysis.

Tabla 32

Resultados del análisis estructural – Modelo 04 (OpenRSE)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.051	0.063	0.000	0.00	0.00	0.00	1	0.938	0.000	0.000
3	0.032	0.141	0.000	0.00	0.00	0.00	3	1.500	0.000	0.000
6	0.000	0.000	0.000	0.00	1.25	0.00	16	0.937	0.000	0.000
8	0.019	0.126	0.000	0.00	0.00	0.00	19	1.688	0.000	0.000
11	0.057	0.076	0.000	0.00	0.00	0.00	-	-	-	-
12	0.064	0.000	0.000	0.00	1.25	0.00	-	-	-	-

Nota. Resultados del análisis del Modelo 04 generados por el software OpenRSE, desarrollado en la presente investigación.

Para validar los resultados de OpenRSE, se presentan las diferencias absolutas, ya que permiten comparaciones directas sin distorsión en magnitudes pequeñas o grandes.

Tabla 33

Errores absolutos – Modelo 04 (OpenRSE vs. SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.000	0.000	0.000	1	0.000	0.000	0.000
3	0.000	0.000	0.000	0.000	0.000	0.000	3	0.000	0.000	0.000
6	0.000	0.000	0.000	0.000	0.000	0.000	16	0.001	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	19	0.000	0.000	0.000
11	0.000	0.000	0.000	0.000	0.000	0.000	-	-	-	-
12	0.000	0.000	0.000	0.000	0.000	0.000	-	-	-	-

Nota. Errores absolutos calculados, basados en la comparación de los resultados de OpenRSE.

Tabla 34

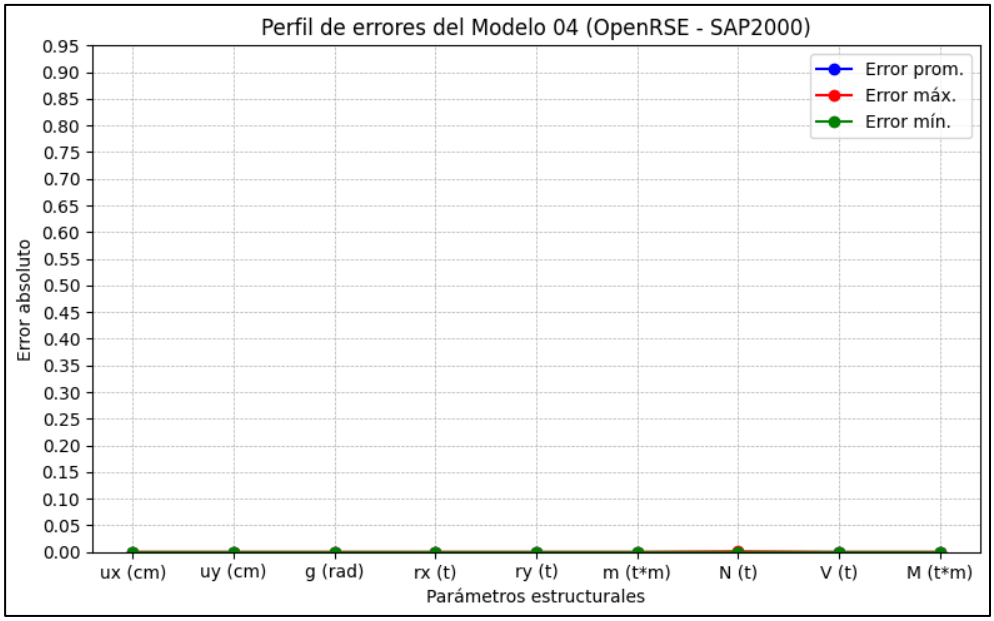
Resumen estadístico de errores absolutos – Modelo 04 (OpenRSE vs. SAP2000)

Medida	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Err. max.	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.000
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Nota. Los resultados evidencian una coincidencia prácticamente exacta, con errores absolutos nulos o despreciables en todos los parámetros estructurales evaluados.

Figura 50

Perfil de errores absolutos – Modelo 04 (OpenRSE vs. SAP2000)



Nota. Todos los errores resultaron cercanos a cero, lo que refleja una alta precisión en los resultados obtenidos por OpenRSE.

Tabla 35

Errores absolutos – Modelo 04 (OpenRSE vs. Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.000	0.000	0.000	1	0.000	0.000	0.000
3	0.000	0.000	0.000	0.000	0.000	0.000	3	0.000	0.000	0.000
6	0.000	0.000	0.000	0.000	0.000	0.000	16	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	19	0.001	0.000	0.000
11	0.000	0.000	0.000	0.000	0.000	0.000	-	-	-	-
12	0.000	0.000	0.000	0.000	0.000	0.000	-	-	-	-

Nota. Errores absolutos calculado, basados en la comparación de los resultados de OpenRSE.

Tabla 36

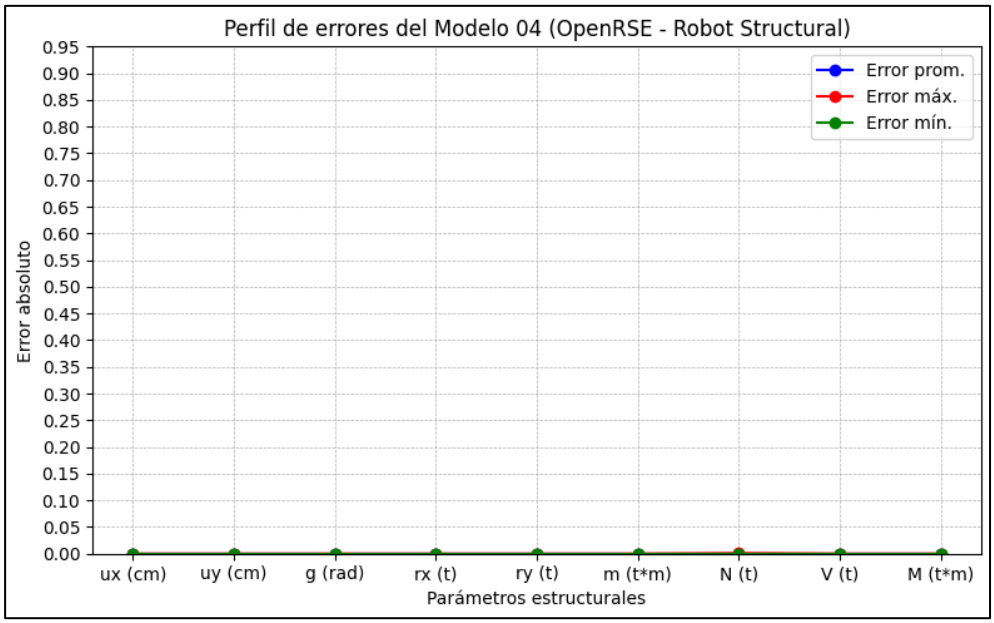
Resumen estadístico de errores absolutos – Modelo 04 (OpenRSE vs. Robot Structural)

Medida	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Err. max.	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.000
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Nota. Los resultados evidencian una coincidencia prácticamente exacta, con errores absolutos nulos o despreciables en todos los parámetros estructurales evaluados.

Figura 51

Perfil de errores absolutos – Modelo 04 (OpenRSE vs. Robot Structural)



Nota. Todos los errores resultaron cercanos a cero, lo que refleja una alta precisión en los resultados obtenidos por OpenRSE.

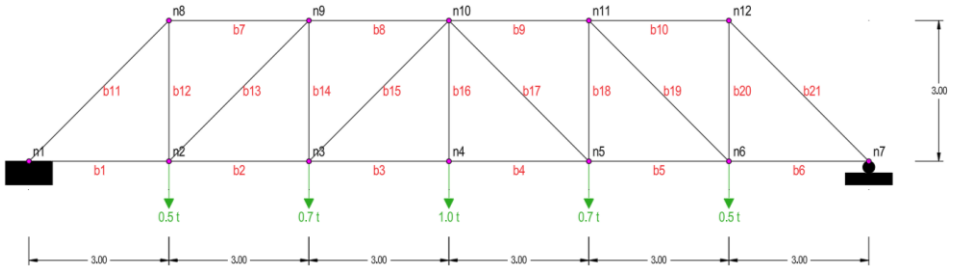
Modelo 05: Estructura de paso

Tabla 37

Características y Propiedades del Modelo 05

Modelo: Estructura de paso	Propiedades físicas:
Tipo: Cercha Pratt	M. de elasticidad: 20389019 t/m²
Numero de barras: 21	Sección: 0.1 x 0.1 x 0.005 m
Numero de nodos: 12	Área: 0.0019 m²
Observaciones: Se usa el sistema internaciones de unidades (SI)	Momento de inercia: 0.00000287 m⁴ (Acero A36)

Representación del Modelo Matemático



Nota. El modelo 05 representa una estructura de paso compuesta por una cercha tipo Pratt, con distribución de cargas puntuales en los nudos inferiores. Este tipo de configuración es representativa de puentes peatonales o pasarelas estructurales, donde se prioriza eficiencia estructural mediante elementos dispuestos para trabajar en tracción y compresión.

A continuación, se presentan los resultados individuales obtenidos en los tres programas de análisis estructural utilizados: SAP2000, Robot Structural y OpenRSE. Para facilitar la revisión, se han seleccionado los nodos y barras más representativos del modelo, permitiendo una lectura concisa y relevante de los parámetros clave como desplazamientos, reacciones y esfuerzos internos.

Tabla 38

Resultados del análisis estructural – Modelo 05 (SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.00	1.70	0.00	1	1.700	0.000	0.000
4	0.062	0.300	0.000	0.00	0.00	0.00	4	3.400	0.000	0.000
7	0.124	0.000	0.000	0.00	1.70	0.00	7	1.700	0.000	0.000
8	0.098	0.135	0.000	0.00	0.00	0.00	9	2.900	0.000	0.000
11	0.039	0.246	0.000	0.00	0.00	0.00	-	-	-	-
12	0.026	0.135	0.000	0.00	0.00	0.00	-	-	-	-

Nota. Resultados del análisis del Modelo 05 generados por el software SAP2000.

Tabla 39

Resultados del análisis estructural – Modelo 05 (Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.00	1.70	0.00	1	1.700	0.000	0.000
4	0.062	0.300	0.000	0.00	0.00	0.00	4	3.400	0.000	0.000
7	0.124	0.000	0.000	0.00	1.70	0.00	7	1.700	0.000	0.000
8	0.098	0.135	0.000	0.00	0.00	0.00	9	2.900	0.000	0.000
11	0.039	0.246	0.000	0.00	0.00	0.00	-	-	-	-
12	0.026	0.135	0.000	0.00	0.00	0.00	-	-	-	-

Nota. Resultados del análisis del Modelo 05 generados por el software Autodesk Robot Structural Analysis.

Tabla 40

Resultados del análisis estructural – Modelo 05 (OpenRSE)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.00	1.70	0.00	1	1.700	0.000	0.000
4	0.062	0.300	0.000	0.00	0.00	0.00	4	3.400	0.000	0.000
7	0.124	0.000	0.000	0.00	1.70	0.00	7	1.700	0.000	0.000
8	0.098	0.135	0.000	0.00	0.00	0.00	9	2.900	0.000	0.000
11	0.040	0.246	0.000	0.00	0.00	0.00	-	-	-	-
12	0.026	0.135	0.000	0.00	0.00	0.00	-	-	-	-

Nota. Resultados del análisis del Modelo 05 generados por el software OpenRSE, desarrollado en la presente investigación.

Para validar los resultados de OpenRSE, se presentan las diferencias absolutas, ya que permiten comparaciones directas sin distorsión en magnitudes pequeñas o grandes.

Tabla 41

Errores absolutos – Modelo 05 (OpenRSE vs. SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.000	0.000	0.000	1	0.000	0.000	0.000
4	0.000	0.000	0.000	0.000	0.000	0.000	4	0.000	0.000	0.000
7	0.000	0.000	0.000	0.000	0.000	0.000	7	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	9	0.000	0.000	0.000
11	0.001	0.000	0.000	0.000	0.000	0.000				
12	0.000	0.000	0.000	0.000	0.000	0.000				

Nota. Errores absolutos calculados, basados en la comparación de los resultados de OpenRSE.

Tabla 42

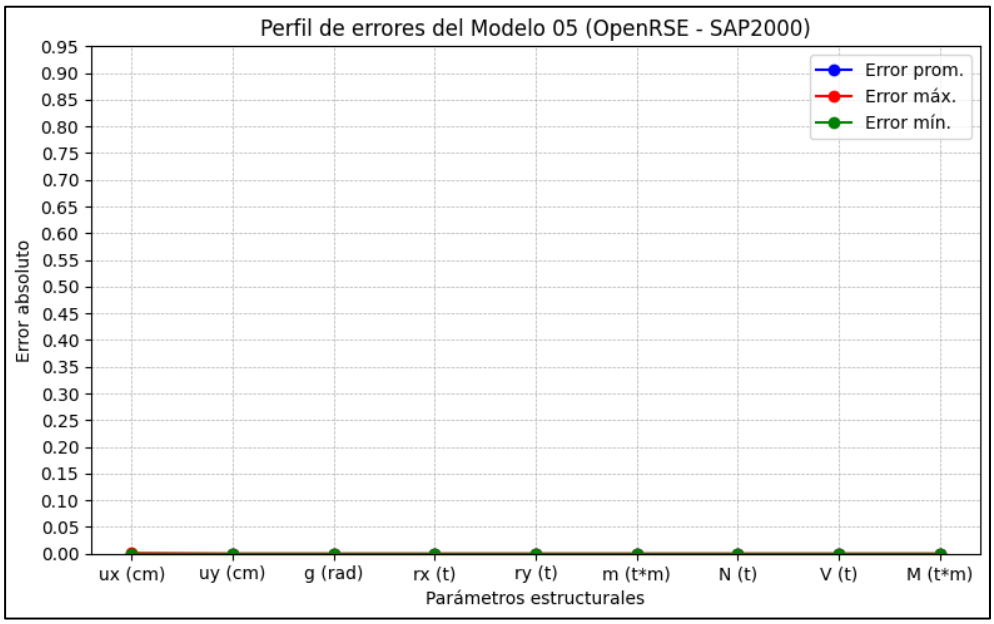
Resumen estadístico de errores absolutos – Modelo 05 (OpenRSE vs. SAP2000)

Medida	Ux (cm)	Uy (cm)	G (rad)	Rx t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Err. max.	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Nota. Los resultados evidencian una coincidencia prácticamente exacta, con errores absolutos nulos o despreciables en todos los parámetros estructurales evaluados.

Figura 52

Perfil de errores absolutos – Modelo 05 (OpenRSE vs. SAP2000)



Nota. Todos los errores resultaron cercanos a cero, lo que refleja una alta precisión en los resultados obtenidos por OpenRSE.

Tabla 43

Errores absolutos – Modelo 05 (OpenRSE vs. Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.000	0.000	0.000	1	0.000	0.000	0.000
4	0.000	0.000	0.000	0.000	0.000	0.000	4	0.000	0.000	0.000
7	0.000	0.000	0.000	0.000	0.000	0.000	7	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	9	0.000	0.000	0.000
11	0.001	0.000	0.000	0.000	0.000	0.000	-	-	-	-
12	0.000	0.000	0.000	0.000	0.000	0.000	-	-	-	-

Nota. Errores absolutos calculado, basados en la comparación de los resultados de OpenRSE.

Tabla 44

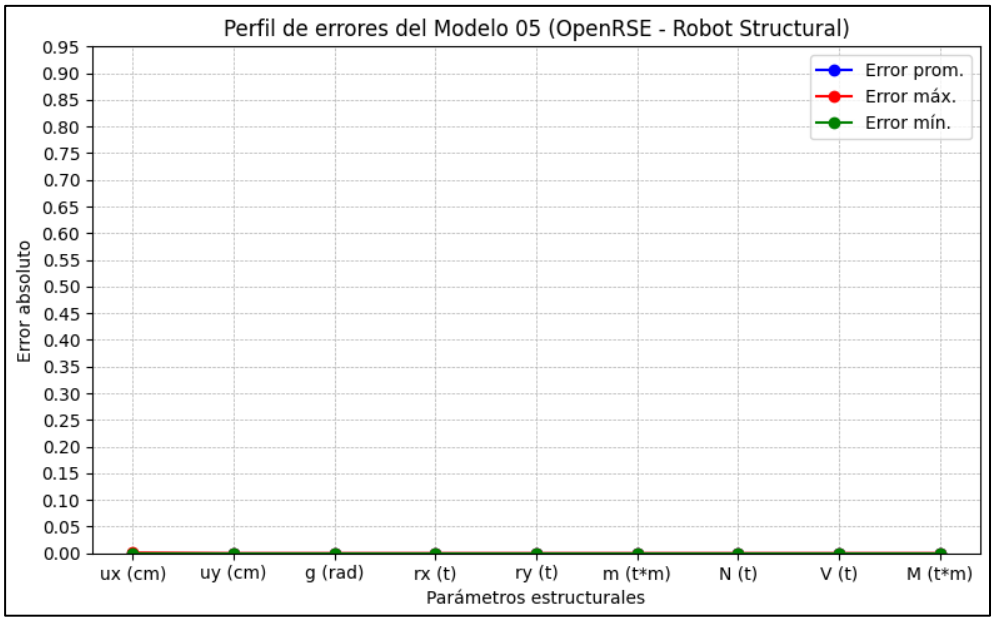
Resumen estadístico de errores absolutos – Modelo 05 (OpenRSE vs. Robot Structural)

Medida	Ux cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Err. max.	0.001	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.000
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Nota. Los resultados evidencian una coincidencia prácticamente exacta, con errores absolutos nulos o despreciables en todos los parámetros estructurales evaluados.

Figura 53

Perfil de errores absolutos – Modelo 05 (OpenRSE vs. Robot Structural)



Nota. Todos los errores resultaron cercanos a cero, lo que refleja una alta precisión en los resultados obtenidos por OpenRSE.

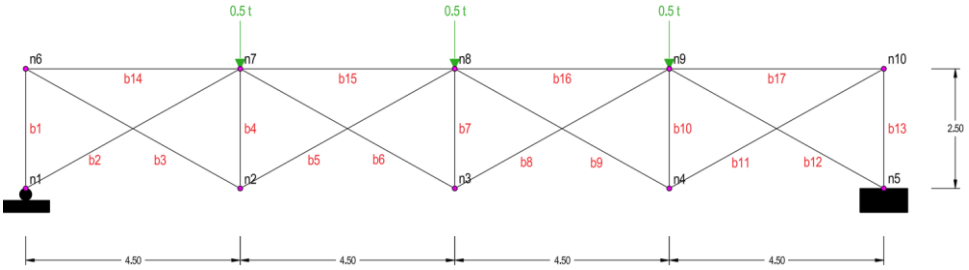
Modelo 06: Pasarela industrial

Tabla 45

Características y Propiedades del Modelo 06

Modelo: Pasarela industrial	Propiedades físicas:
Tipo: Cercha Warren	M. de elasticidad: 20389019 t/m²
Numero de barras: 17	Sección: 0.1 x 0.1 x 0.005 m
Numero de nodos: 10	Área: 0.0019 m²
Observaciones: Se usa el sistema internaciones de unidades (SI)	Momento de inercia: 0.00000287 m⁴ (Acero A36)

Representación del Modelo Matemático



Nota. El modelo 06 corresponde a una pasarela estructural compuesta por una cercha tipo Warren, con disposición triangular alternada que permite una distribución eficiente de esfuerzos. Este tipo de configuración es común en aplicaciones industriales y sistemas de paso elevados, donde se requiere simplicidad constructiva y resistencia estructural.

A continuación, se presentan los resultados individuales obtenidos en los tres programas de análisis estructural utilizados: SAP2000, Robot Structural y OpenRSE. Para facilitar la revisión, se han seleccionado los nodos y barras más representativos del modelo, permitiendo una lectura concisa y relevante de los parámetros clave como desplazamientos, reacciones y esfuerzos internos.

Tabla 46

Resultados del análisis estructural – Modelo 06 (SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.302	0.000	0.000	0.00	0.75	0.00	1	0.750	0.000	0.000
3	0.151	0.460	0.000	0.00	0.00	0.00	2	0.000	0.000	0.000
5	0.000	0.000	0.000	0.00	0.75	0.00	6	2.059	0.000	0.000
6	0.099	0.005	0.000	0.00	0.00	0.00	7	2.000	0.000	0.000
9	0.188	0.338	0.000	0.00	0.00	0.00	12	0.000	0.000	0.000
-	-	-	-	-	-	-	15	3.150	0.000	0.000
-	-	-	-	-	-	-	17	1.350	0.000	0.000

Nota. Resultados del análisis del Modelo 06 generados por el software SAP2000.

Tabla 47

Resultados del análisis estructural – Modelo 06 (Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.302	0.000	0.000	0.00	0.75	0.00	1	0.750	0.000	0.000
3	0.151	0.460	0.000	0.00	0.00	0.00	2	0.000	0.000	0.000
5	0.000	0.000	0.000	0.00	0.75	0.00	6	2.059	0.000	0.000
6	0.099	0.005	0.000	0.00	0.00	0.00	7	2.000	0.000	0.000
9	0.188	0.338	0.000	0.00	0.00	0.00	12	0.000	0.000	0.000
-	-	-	-	-	-	-	15	3.150	0.000	0.000
-	-	-	-	-	-	-	17	1.350	0.000	0.000

Nota. Resultados del análisis del Modelo 06 generados por el software Autodesk Robot Structural Analysis.

Tabla 48

Resultados del análisis estructural – Modelo 06 (OpenRSE)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.302	0.000	0.000	0.00	0.75	0.00	1	0.750	0.000	0.000
3	0.151	0.460	0.000	0.00	0.00	0.00	2	0.000	0.000	0.000
5	0.000	0.000	0.000	0.00	0.75	0.00	6	2.059	0.000	0.000
6	0.099	0.005	0.000	0.00	0.00	0.00	7	2.000	0.000	0.000
9	0.188	0.338	0.000	0.00	0.00	0.00	12	0.000	0.000	0.000
-	-	-	-	-	-	-	15	3.150	0.000	0.000
-	-	-	-	-	-	-	17	1.350	0.000	0.000

Nota. Resultados del análisis del Modelo 06 generados por el software OpenRSE, desarrollado en la presente investigación.

Para validar los resultados de OpenRSE, se presentan las diferencias absolutas, ya que permiten comparaciones directas sin distorsión en magnitudes pequeñas o grandes.

Tabla 49

Errores absolutos – Modelo 06 (OpenRSE vs. SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.000	0.000	0.000	1	0.000	0.000	0.000
3	0.000	0.000	0.000	0.000	0.000	0.000	2	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000	0.000	6	0.000	0.000	0.000
6	0.000	0.000	0.000	0.000	0.000	0.000	7	0.000	0.000	0.000
9	0.000	0.000	0.000	0.000	0.000	0.000	12	0.000	0.000	0.000
-	-	-	-	-	-	-	15	0.000	0.000	0.000
-	-	-	-	-	-	-	17	0.000	0.000	0.000

Nota. Errores absolutos calculados, basados en la comparación de los resultados de OpenRSE.

Tabla 50

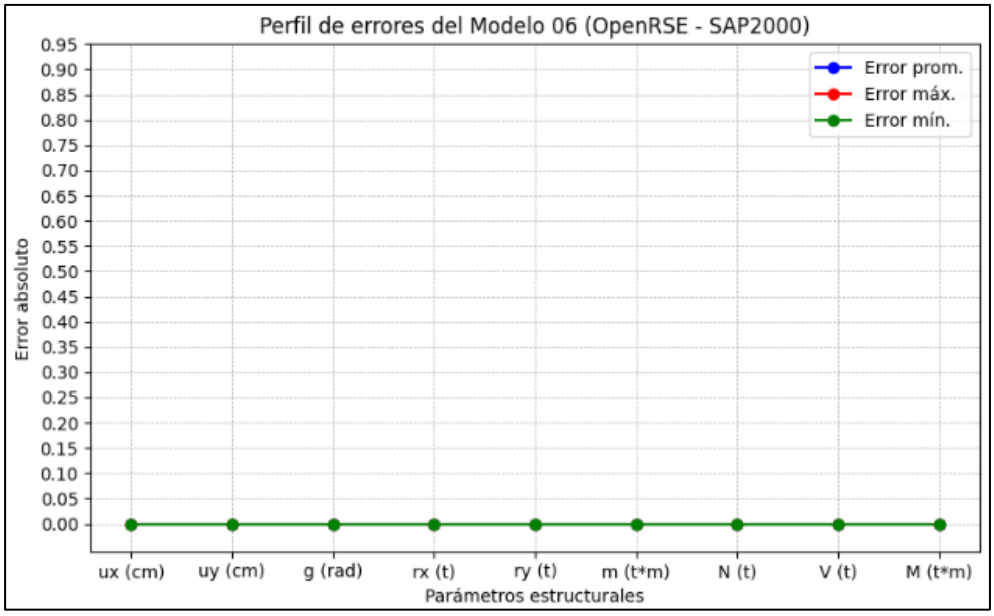
Resumen estadístico de errores absolutos – Modelo 06 (OpenRSE vs. SAP2000)

Medida	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Err. max.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Nota. Los resultados evidencian una coincidencia prácticamente exacta, con errores absolutos nulos o despreciables en todos los parámetros estructurales evaluados.

Figura 54

Perfil de errores absolutos – Modelo 06 (OpenRSE vs. SAP2000)



Nota. Todos los errores resultaron cercanos a cero, lo que refleja una alta precisión en los resultados obtenidos por OpenRSE.

Tabla 51

Errores absolutos – Modelo 06 (OpenRSE vs. Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.000	0.000	0.000	1	0.000	0.000	0.000
3	0.000	0.000	0.000	0.000	0.000	0.000	2	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000	0.000	6	0.000	0.000	0.000
6	0.000	0.000	0.000	0.000	0.000	0.000	7	0.000	0.000	0.000
9	0.000	0.000	0.000	0.000	0.000	0.000	12	0.000	0.000	0.000
-	-	-	-	-	-	-	15	0.000	0.000	0.000
-	-	-	-	-	-	-	17	0.000	0.000	0.000

Nota. Errores absolutos calculado, basados en la comparación de los resultados de OpenRSE.

Tabla 52

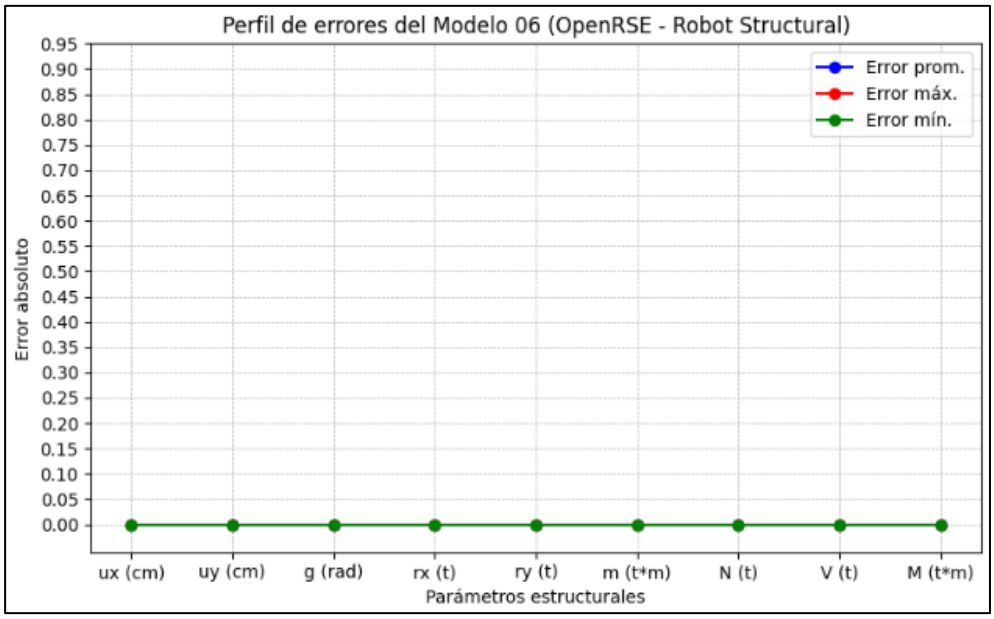
Resumen estadístico de errores absolutos – Modelo 06(OpenRSE vs. Robot Structural)

Medida	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Err. max.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Nota. Los resultados evidencian una coincidencia prácticamente exacta, con errores absolutos nulos o despreciables en todos los parámetros estructurales evaluados.

Figura 55

Perfil de errores absolutos – Modelo 06 (OpenRSE vs. Robot Structural)



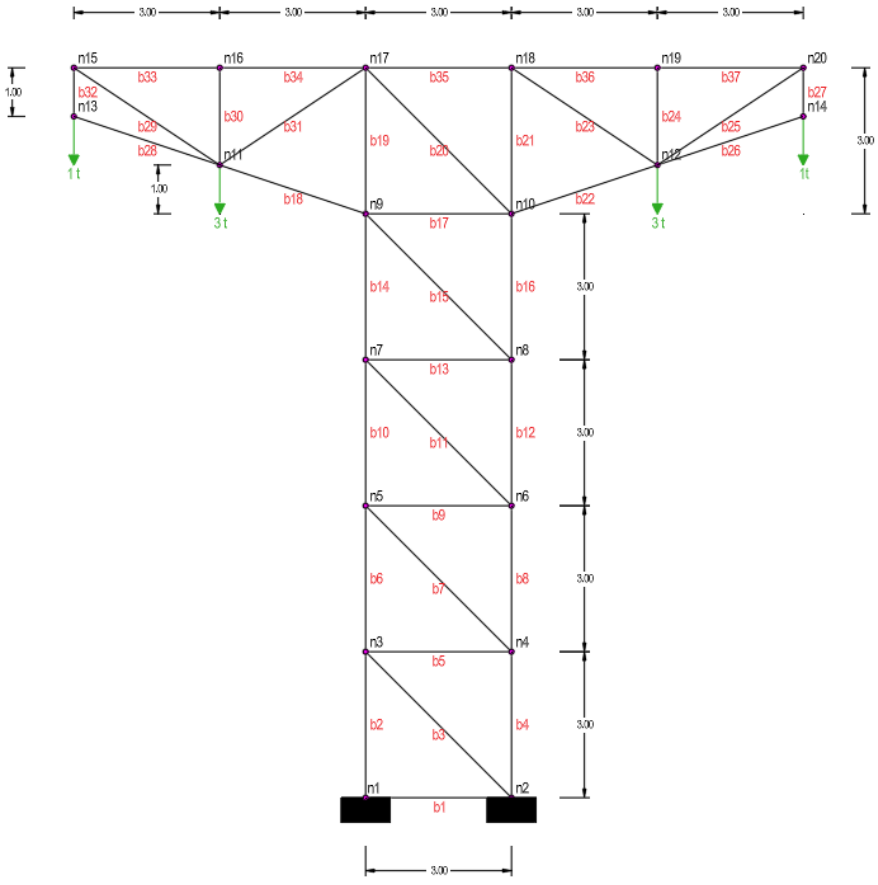
Nota. Todos los errores resultaron cercanos a cero, lo que refleja una alta precisión en los resultados obtenidos por OpenRSE.

Modelo 07: Soporte estructural para grúas

Tabla 53

Características y Propiedades del Modelo 07

Modelo: Soporte estructural para grúas Tipo: Torre estructural con brazo reticulado tipo voladizo Numero de barras: 37 Numero de nodos: 20 Observaciones: Se usa el sistema internaciones de unidades (SI)	Propiedades físicas: M. de elasticidad: 20389019 t/m² Sección: 0.07 x 0.07 x 0.004 m Área: 0.001056 m² Momento de inercia: 0.000000769 m⁴ (Acero A36)
Representación del Modelo Matemático	



Nota. El modelo 07 representa una estructura vertical tipo torre con brazo reticulado en voladizo, común en aplicaciones industriales como soportes para grúas o mástiles técnicos. Su configuración permite el análisis del equilibrio estructural frente a cargas aplicadas a diferentes niveles, evaluando el comportamiento en sistemas compuestos por elementos inclinados y verticales.

A continuación, se presentan los resultados individuales obtenidos en los tres programas de análisis estructural utilizados: SAP2000, Robot Structural y OpenRSE. Para facilitar la revisión, se han seleccionado los nodos y barras más representativos del modelo, permitiendo una lectura concisa y relevante de los parámetros clave como desplazamientos, reacciones y esfuerzos internos.

Tabla 54

Resultados del análisis estructural – Modelo 07 (SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.00	4.00	0.00	1	0.000	0.000	0.000
2	0.000	0.000	0.000	0.00	4.00	0.00	6	4.000	0.000	0.000
5	0.111	0.111	0.000	0.00	0.00	0.00	16	4.000	0.000	0.000
10	0.293	0.223	0.000	0.00	0.00	0.00	17	5.000	0.000	0.000
11	0.238	0.513	0.000	0.00	0.00	0.00	18	5.270	0.000	0.000
14	0.237	0.635	0.000	0.00	0.00	0.00	26	0.000	0.000	0.000
16	0.346	0.513	0.000	0.00	0.00	0.00	31	4.207	0.000	0.000

Nota. Resultados del análisis del Modelo 07 generados por el software SAP2000.

Tabla 55

Resultados del análisis estructural – Modelo 07 (Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.00	4.00	0.00	1	0.000	0.000	0.000
2	0.000	0.000	0.000	0.00	4.00	0.00	6	4.000	0.000	0.000
5	0.111	0.111	0.000	0.00	0.00	0.00	16	4.000	0.000	0.000
10	0.293	0.223	0.000	0.00	0.00	0.00	17	5.000	0.000	0.000
11	0.238	0.513	0.000	0.00	0.00	0.00	18	5.270	0.000	0.000
14	0.237	0.635	0.000	0.00	0.00	0.00	26	0.000	0.000	0.000
16	0.346	0.513	0.000	0.00	0.00	0.00	31	4.206	0.000	0.000

Nota. Resultados del análisis del Modelo 07 generados por el software Autodesk Robot Structural Analysis.

Tabla 56

Resultados del análisis estructural – Modelo 07 (OpenRSE)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.00	4.00	0.00	1	0.000	0.000	0.000
2	0.000	0.000	0.000	0.00	4.00	0.00	6	4.000	0.000	0.000
5	0.111	0.111	0.000	0.00	0.00	0.00	16	4.000	0.000	0.000
10	0.293	0.223	0.000	0.00	0.00	0.00	17	5.000	0.000	0.000
11	0.238	0.513	0.000	0.00	0.00	0.00	18	5.270	0.000	0.000
14	0.237	0.635	0.000	0.00	0.00	0.00	26	0.000	0.000	0.000
16	0.346	0.513	0.000	0.00	0.00	0.00	31	4.206	0.000	0.000

Nota. Resultados del análisis del Modelo 07 generados por el software OpenRSE, desarrollado en la presente investigación.

Para validar los resultados de OpenRSE, se presentan las diferencias absolutas, ya que permiten comparaciones directas sin distorsión en magnitudes pequeñas o grandes.

Tabla 57

Errores absolutos – Modelo 07 (OpenRSE vs. SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.000	0.000	0.000	1	0.000	0.000	0.000
2	0.000	0.000	0.000	0.000	0.000	0.000	6	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000	0.000	16	0.000	0.000	0.000
10	0.000	0.000	0.000	0.000	0.000	0.000	17	0.000	0.000	0.000
11	0.000	0.000	0.000	0.000	0.000	0.000	18	0.000	0.000	0.000
14	0.000	0.000	0.000	0.000	0.000	0.000	26	0.000	0.000	0.000
16	0.000	0.000	0.000	0.000	0.000	0.000	31	0.001	0.000	0.000

Nota. Errores absolutos calculados, basados en la comparación de los resultados de OpenRSE.

Tabla 58

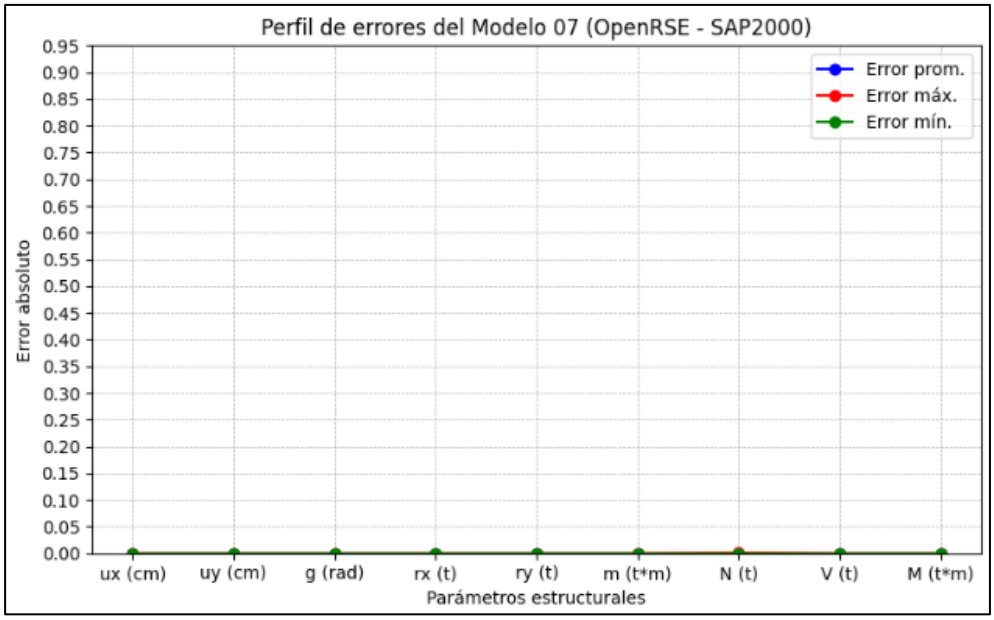
Resumen estadístico de errores absolutos – Modelo 07 (OpenRSE vs. SAP2000)

Medida	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Err. max.	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.000
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Nota. Los resultados evidencian una coincidencia prácticamente exacta, con errores absolutos nulos o despreciables en todos los parámetros estructurales evaluados.

Figura 56

Perfil de errores absolutos – Modelo 07 (OpenRSE vs. SAP2000)



Nota. Todos los errores resultaron cercanos a cero, lo que refleja una alta precisión en los resultados obtenidos por OpenRSE.

Tabla 59

Errores absolutos – Modelo 07 (OpenRSE vs. Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.000	0.000	0.000	1	0.000	0.000	0.000
2	0.000	0.000	0.000	0.000	0.000	0.000	6	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000	0.000	16	0.000	0.000	0.000
10	0.000	0.000	0.000	0.000	0.000	0.000	17	0.000	0.000	0.000
11	0.000	0.000	0.000	0.000	0.000	0.000	18	0.000	0.000	0.000
14	0.000	0.000	0.000	0.000	0.000	0.000	26	0.000	0.000	0.000
16	0.000	0.000	0.000	0.000	0.000	0.000	31	0.000	0.000	0.000

Nota. Errores absolutos calculado, basados en la comparación de los resultados de OpenRSE.

Tabla 60

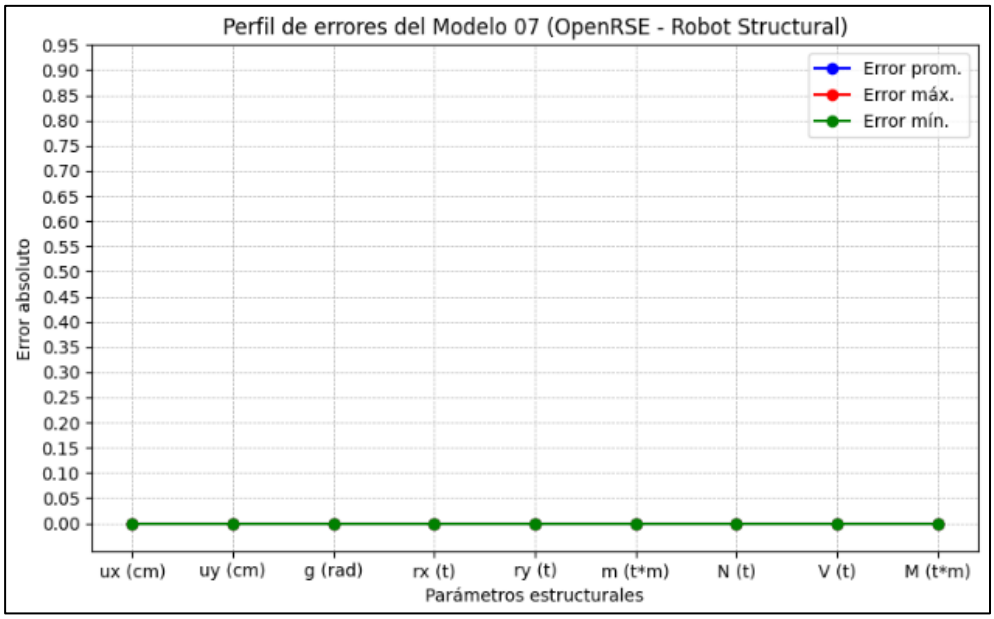
Resumen estadístico de errores absolutos – Modelo 07 (OpenRSE vs. Robot Structural)

13	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Err. max.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Nota. Los resultados evidencian una coincidencia prácticamente exacta, con errores absolutos nulos o despreciables en todos los parámetros estructurales evaluados.

Figura 57

Perfil de errores absolutos – Modelo 07 (OpenRSE vs. Robot Structural)



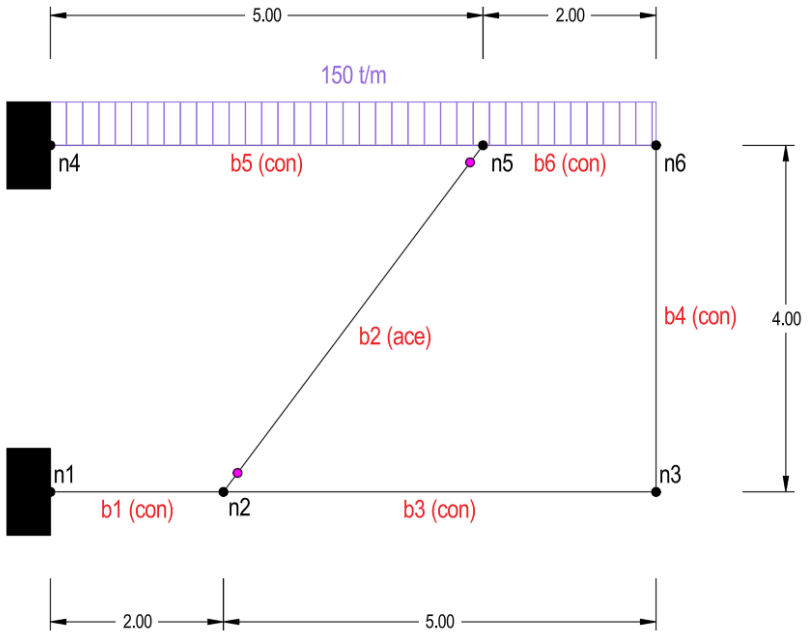
Nota. Todos los errores resultaron cercanos a cero, lo que refleja una alta precisión en los resultados obtenidos por OpenRSE.

Modelo 08: Estructura de soporte

Tabla 61

Características y Propiedades del Modelo 08

Modelo: Estructura de soporte Tipo: Marco asimétrico reforzado con diagonal interior Numero de barras: 6 Numero de nodos: 6 Observaciones: Se usa el sistema internaciones de unidades (SI)	Propiedades físicas: Acero A36 con E = 20389019 t/m², Sec. = 0.10 x 0.10 x 0.015 m, A = 0.0051 m², I = 0.0000063325 m⁴; y Concreto con E = 2173706.5 t/m², Sec. = 0.50 x 0.50 m, A = 0.25 m², I = 0.0052083 m⁴.
Representación del Modelo Matemático	



Nota. El modelo 08 representa un marco estructural asimétrico con refuerzo mediante una barra diagonal interior, diseñado con elementos de concreto y aceros combinados. Su configuración permite evaluar la transferencia de cargas en condiciones desbalanceadas, siendo útil en estructuras técnicas que requieren refuerzo puntual.

A continuación, se presentan los resultados individuales obtenidos en los tres programas de análisis estructural utilizados: SAP2000, Robot Structural y OpenRSE. Para facilitar la revisión, se han seleccionado los nodos y barras más representativos del modelo, permitiendo una lectura concisa y relevante de los parámetros clave como desplazamientos, reacciones y esfuerzos internos.

Tabla 62

Resultados del análisis estructural – Modelo 08 (SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	534.08	592.86	949.46	1	534.078	592.861	949.463
2	0.197	10.419	0.063	0.00	0.00	0.00	2	785.445	0.000	0.000
3	0.254	22.269	0.002	0.00	0.00	0.00	3	62.811	35.495	236.259
4	0.000	0.000	0.000	534.08	457.14	589.23	4	35.495	62.811	192.461
5	0.491	15.656	0.032	0.00	0.00	0.00	5	534.078	457.139	589.225
6	0.515	22.243	0.021	0.00	0.00	0.00	6	62.811	335.495	192.461

Nota. Resultados del análisis del Modelo 08 generados por el software SAP2000.

Tabla 63

Resultados del análisis estructural – Modelo 08 (Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	535.82	595.14	949.92	1	535.821	595.144	949.919
2	0.197	9.772	0.063	0.00	0.00	0.00	2	790.043	0.000	0.000
3	0.254	21.358	0.003	0.00	0.00	0.00	3	61.795	36.890	240.369
4	0.000	0.000	0.000	535.82	454.86	581.80	4	36.890	61.795	191.265
5	0.493	15.038	0.031	0.00	0.00	0.00	5	535.821	454.856	581.795
6	0.516	21.331	0.021	0.00	0.00	0.00	6	61.795	336.890	191.265

Nota. Resultados del análisis del Modelo 08 generados por el software Autodesk Robot Structural Analysis.

Tabla 64

Resultados del análisis estructural – Modelo 08 (OpenRSE)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	535.82	595.15	949.92	1	535.822	595.145	949.919
2	0.197	9.772	0.063	0.00	0.00	0.00	2	790.044	0.000	0.000
3	0.254	21.358	0.003	0.00	0.00	0.00	3	61.795	36.891	240.370
4	0.000	0.000	0.000	535.82	454.86	581.79	4	36.891	61.795	191.264
5	0.493	15.038	0.031	0.00	0.00	0.00	5	535.822	454.855	581.794
6	0.516	21.331	0.021	0.00	0.00	0.00	6	61.795	336.891	191.264

Nota. Resultados del análisis del Modelo 08 generados por el software OpenRSE, desarrollado en la presente investigación.

Para validar los resultados de OpenRSE, se presentan las diferencias absolutas, ya que permiten comparaciones directas sin distorsión en magnitudes pequeñas o grandes.

Tabla 65

Errores absolutos – Modelo 08 (OpenRSE vs. SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	1.742	2.285	0.459	1	1.744	2.284	0.456
2	0.000	0.647	0.000	0.000	0.000	0.000	2	4.599	0.000	0.000
3	0.000	0.911	0.001	0.000	0.000	0.000	3	1.016	1.396	4.111
4	0.000	0.000	0.000	1.742	2.285	7.436	4	1.396	1.016	1.197
5	0.002	0.618	0.001	0.000	0.000	0.000	5	1.744	2.284	7.431
6	0.001	0.912	0.000	0.000	0.000	0.000	6	1.016	1.396	1.197

Nota. Errores absolutos calculados, basados en la comparación de los resultados de OpenRSE.

Tabla 66

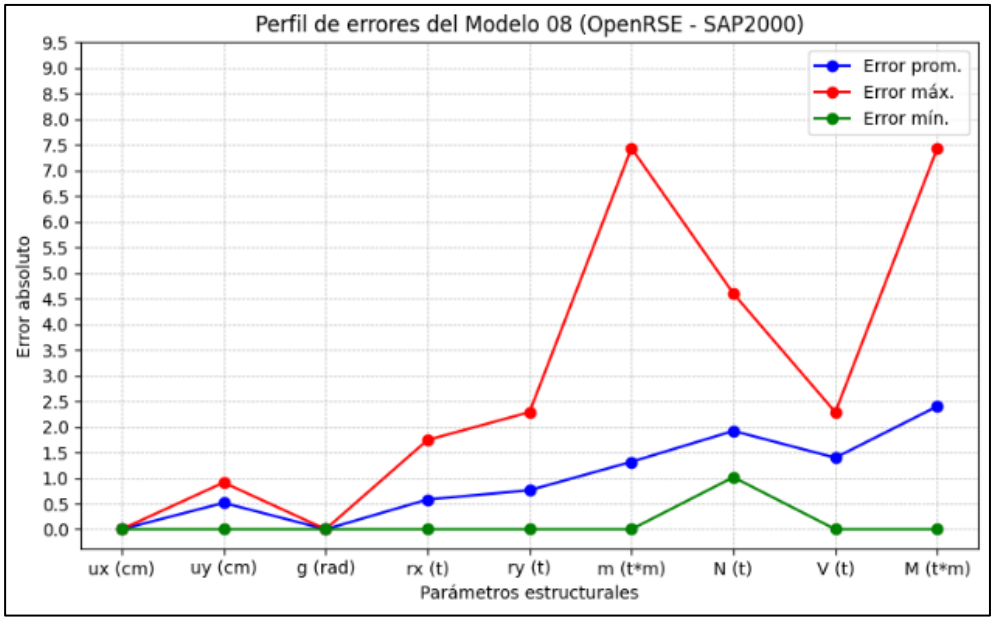
Resumen estadístico de errores absolutos – Modelo 08 (OpenRSE vs. SAP2000)

Medida	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.001	0.515	0.000	0.581	0.762	1.316	1.919	1.396	2.399
Err. max.	0.002	0.912	0.001	1.742	2.285	7.436	4.599	2.284	7.431
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	1.016	0.000	0.000

Nota. El mayor error registrado corresponde al momento flector (7.436 t*m), seguido por la normal (4.599 t), ambos dentro de márgenes aceptables para análisis estructurales considerando las grandes cargas, sin comprometer la precisión del modelo.

Figura 58

Perfil de errores absolutos – Modelo 08 (OpenRSE vs. SAP2000)



Nota. El gráfico evidencia que los errores máximos se concentran principalmente en los momentos flectores, mientras que en la mayoría de los parámetros los errores mínimos son nulos y los errores promedio se mantienen bajos.

Tabla 67

Errores absolutos – Modelo 08 (OpenRSE vs. Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.001	0.001	0.000	1	0.001	0.001	0.000
2	0.000	0.000	0.000	0.000	0.000	0.000	2	0.001	0.000	0.000
3	0.000	0.000	0.000	0.000	0.000	0.000	3	0.000	0.001	0.001
4	0.000	0.000	0.000	0.001	0.001	0.001	4	0.001	0.000	0.001
5	0.000	0.000	0.000	0.000	0.000	0.000	5	0.001	0.001	0.001
6	0.000	0.000	0.000	0.000	0.000	0.000	6	0.000	0.001	0.001

Nota. Errores absolutos calculado, basados en la comparación de los resultados de OpenRSE.

Tabla 68

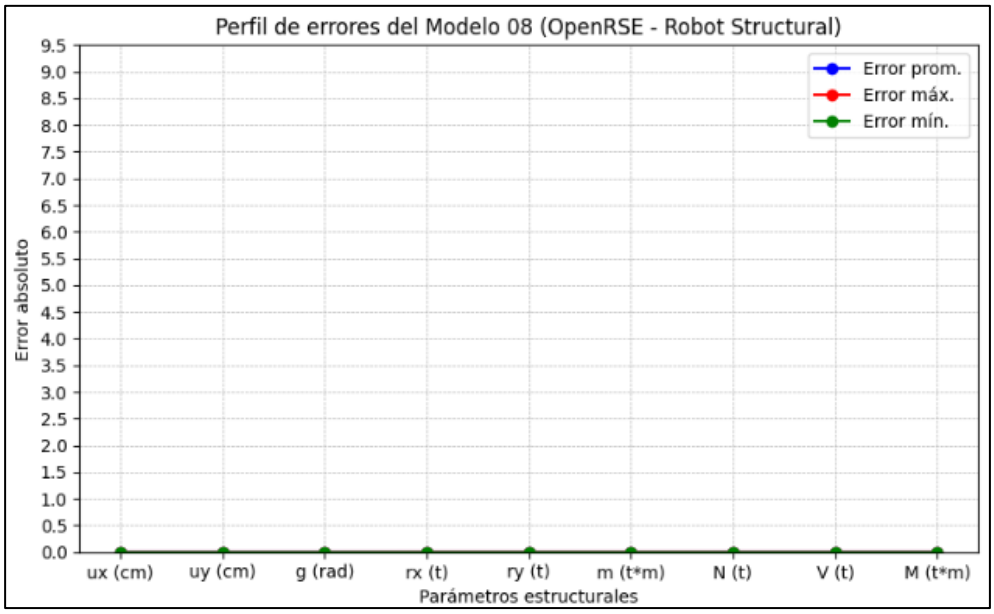
Resumen estadístico de errores absolutos – Modelo 08 (OpenRSE vs. Robot Structural)

Medida	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.001
Err. max.	0.000	0.000	0.000	0.001	0.001	0.001	0.001	0.001	0.001
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Nota. Los resultados evidencian una coincidencia prácticamente exacta, con errores absolutos nulos o despreciables en todos los parámetros estructurales evaluados.

Figura 59

Perfil de errores absolutos – Modelo 08 (OpenRSE vs. Robot Structural)

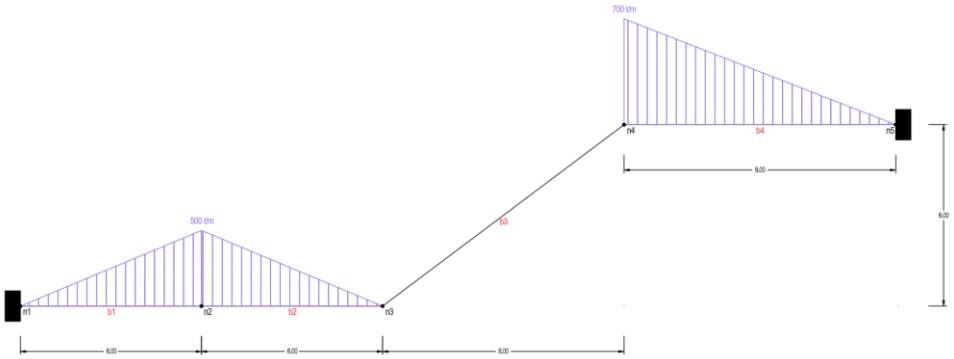


Nota. Todos los errores resultaron cercanos a cero, lo que refleja una alta precisión en los resultados obtenidos por OpenRSE.

Modelo 09: Pasarela o rampa estructural

Tabla 69

Características y Propiedades del Modelo 09

Modelo: Pasarela o rampa estructural Tipo: Viga continua de concreto armado con cambio de nivel Numero de barras: 4 Numero de nodos: 5 Observaciones: Se usa el sistema internaciones de unidades (SI)	Propiedades físicas: M. de elasticidad: 2173706.5 t/m² Sección: 1.0 x 1.0 m Área: 1.0 m² Momento de inercia: 0.0833333 m⁴ (Concreto f'c 210 kg/cm2)
Representación del Modelo Matemático	
	

Nota. El modelo 09 representa una viga continua de concreto armado con cambio de nivel, sometida a cargas triangulares distribuidas. Esta configuración es útil para simular rampas estructurales, pasarelas técnicas o sistemas de conexión entre plataformas con diferentes elevaciones. El diseño permite estudiar los efectos del desnivel en estructuras monolíticas.

A continuación, se presentan los resultados individuales obtenidos en los tres programas de análisis estructural utilizados: SAP2000, Robot Structural y OpenRSE. Para facilitar la revisión, se han seleccionado los nodos y barras más representativos del modelo, permitiendo una lectura concisa y relevante de los parámetros clave como desplazamientos, reacciones y esfuerzos internos.

Tabla 70

Resultados del análisis estructural – Modelo 09 (SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	1084.20	2678.05	12148.40	1	1084.197	2678.045	12148.400
2	0.299	72.208	0.161	0.00	0.00	0.00	2	1084.197	1178.045	2579.005
3	0.599	148.393	0.088	0.00	0.00	0.00	3	1060.531	392.954	5917.682
4	0.449	146.183	0.130	0.00	0.00	0.00	4	1084.197	3471.955	15879.913
5	0.000	0.000	0.000	1084.20	3471.96	15879.91	-	-	-	-

Nota. Resultados del análisis del Modelo 09 generados por el software SAP2000.

Tabla 71

Resultados del análisis estructural – Modelo 09 (Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	1111.18	2669.39	12095.21	1	1111.179	2669.392	12095.207
2	0.307	70.119	0.160	0.00	0.00	0.00	2	1111.179	1169.392	2557.967
3	0.613	145.780	0.088	0.00	0.00	0.00	3	1087.308	402.221	5959.708
4	0.460	143.515	0.130	0.00	0.00	0.00	4	1111.179	3480.608	15915.765
5	0.000	0.000	0.000	1111.18	3480.61	15915.77	-	-	-	-

Nota. Resultados del análisis del Modelo 09 generados por el software Autodesk Robot Structural Analysis.

Tabla 72

Resultados del análisis estructural – Modelo 09 (OpenRSE)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	1111.18	2669.39	12095.21	1	1111.179	2669.392	12095.207
2	0.307	70.119	0.160	0.00	0.00	0.00	2	1111.179	1169.392	2557.967
3	0.613	145.781	0.088	0.00	0.00	0.00	3	1087.308	402.221	5959.708
4	0.460	143.516	0.130	0.00	0.00	0.00	4	1111.179	3480.608	15915.765
5	0.000	0.000	0.000	1111.18	3480.61	15915.77	-	-	-	-

Nota. Resultados del análisis del Modelo 09 generados por el software OpenRSE, desarrollado en la presente investigación.

Para validar los resultados de OpenRSE, se presentan las diferencias absolutas, ya que permiten comparaciones directas sin distorsión en magnitudes pequeñas o grandes.

Tabla 73

Errores absolutos – Modelo 09 (OpenRSE vs. SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	26.982	8.653	53.193	1	26.982	8.653	53.193
2	0.008	2.089	0.001	0.000	0.000	0.000	2	26.982	8.653	21.038
3	0.014	2.612	0.000	0.000	0.000	0.000	3	26.777	9.267	42.026
4	0.011	2.667	0.000	0.000	0.000	0.000	4	26.982	8.653	35.852
5	0.000	0.000	0.000	26.982	8.653	35.852	-	-	-	-

Nota. Errores absolutos calculados, basados en la comparación de los resultados de OpenRSE.

Tabla 74

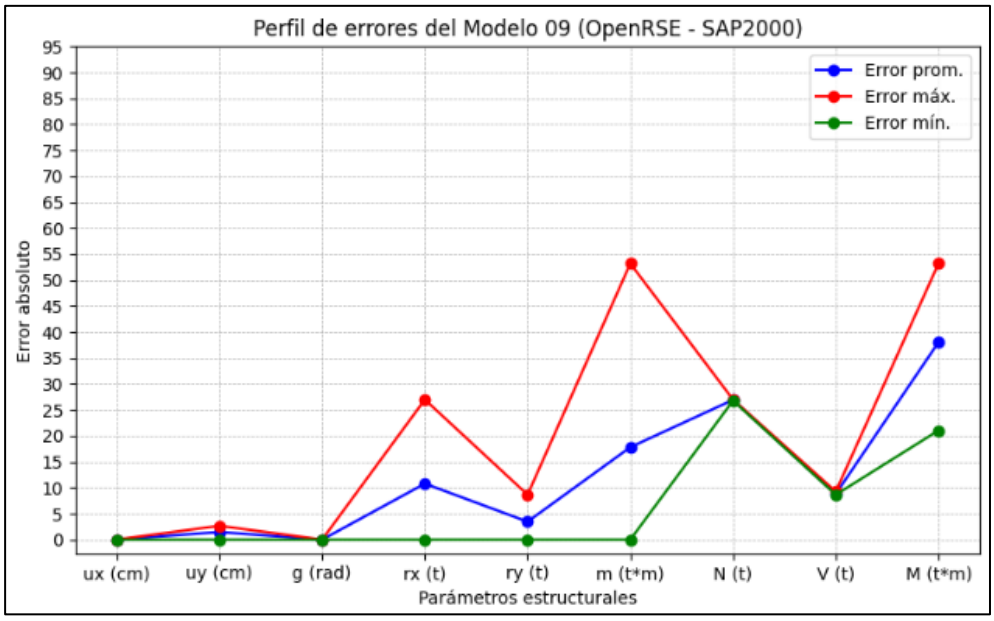
Resumen estadístico de errores absolutos – Modelo 09 (OpenRSE vs. SAP2000)

Medida	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.007	1.474	0.000	10.793	3.461	17.809	26.931	8.807	38.027
Err. max.	0.014	2.667	0.001	26.982	8.653	53.193	26.982	9.267	53.193
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	26.777	8.653	21.038

Nota. El mayor error registrado corresponde al momento flector (53.193 t*m), seguido por la normal (26.982 t), ambos dentro de márgenes aceptables para análisis estructurales considerando las grandes cargas, sin comprometer la precisión del modelo.

Figura 60

Perfil de errores absolutos – Modelo 09 (OpenRSE vs. SAP2000)



Nota. El gráfico evidencia que los errores máximos se concentran principalmente en los momentos flectores, mientras que en la mayoría de los parámetros los errores mínimos son nulos y los errores promedio se mantienen bajos considerando las grandes cargas aplicadas en el modelo.

Tabla 75

Errores absolutos – Modelo 09 (OpenRSE vs. Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.000	0.000	0.000	1	0.000	0.000	0.000
2	0.000	0.000	0.000	0.000	0.000	0.000	2	0.000	0.000	0.000
3	0.000	0.001	0.000	0.000	0.000	0.000	3	0.000	0.000	0.000
4	0.000	0.001	0.000	0.000	0.000	0.000	4	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000	0.000	-	-	-	-

Nota. Errores absolutos calculado, basados en la comparación de los resultados de OpenRSE.

Tabla 76

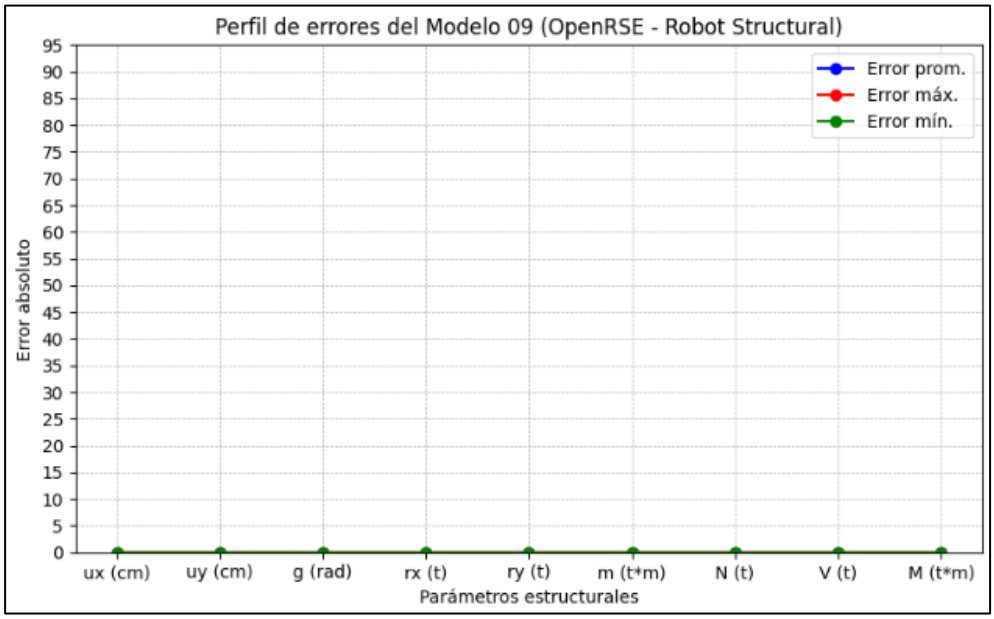
Resumen estadístico de errores absolutos – Modelo 09 (OpenRSE vs. Robot Structural)

Medida	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Err. max.	0.000	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Nota. Los resultados evidencian una coincidencia prácticamente exacta, con errores absolutos nulos o despreciables en todos los parámetros estructurales evaluados.

Figura 61

Perfil de errores absolutos – Modelo 09 (OpenRSE vs. Robot Structural)



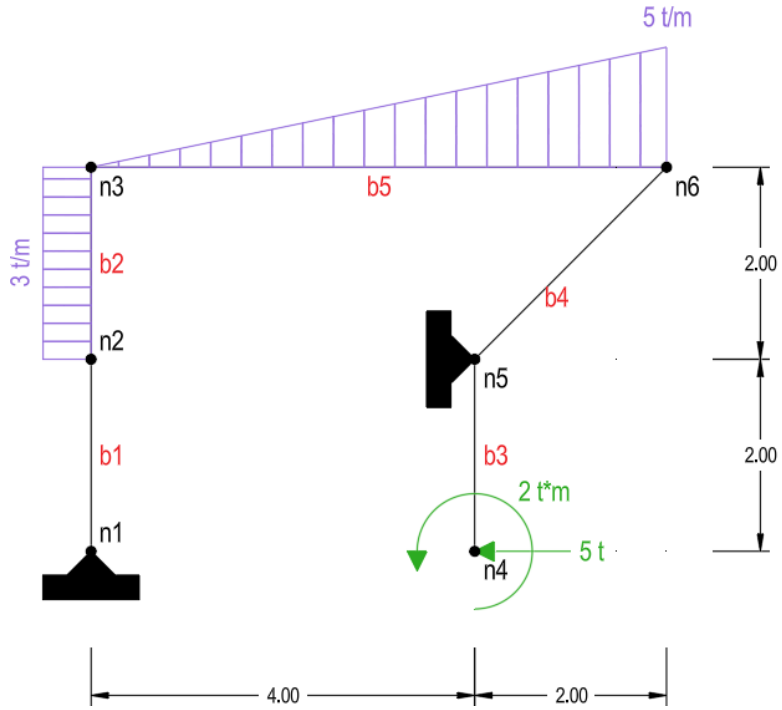
Nota. Todos los errores resultaron cercanos a cero, lo que refleja una alta precisión en los resultados obtenidos por OpenRSE.

Modelo 10: Sistema estructural académico

Tabla 77

Características y Propiedades del Modelo 10

Modelo: Sistema estructural académico Tipo: Marco no articulado con viga inclinada y momento aplicado Numero de barras: 5 Numero de nodos: 6 Observaciones: Se usa el sistema internaciones de unidades (SI)	Propiedades físicas: M. de elasticidad: 2173706.5 t/m² Sección: 0.4 x 0.4 m Área: 0.160 m² Momento de inercia: 0.002133 m⁴ (Concreto f'c 210 kg/cm2)
Representación del Modelo Matemático	



Nota. El modelo 10 representa un marco estructural no articulado con viga inclinada y aplicación de momento externo, desarrollado con fines académicos para evaluar el comportamiento frente a cargas combinadas y desequilibrio estructural. Esta configuración permite estudiar la transferencia de cargas en sistemas asimétricos

A continuación, se presentan los resultados individuales obtenidos en los tres programas de análisis estructural utilizados: SAP2000, Robot Structural y OpenRSE. Para facilitar la revisión, se han seleccionado los

nodos y barras más representativos del modelo, permitiendo una lectura concisa y relevante de los parámetros clave como desplazamientos, reacciones y esfuerzos internos.

Tabla 78

Resultados del análisis estructural – Modelo 10 (SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.022	5.56	6.28	0.00	1	6.281	5.561	11.123
2	4.210	0.004	0.019	0.00	0.00	0.00	2	6.281	5.561	16.245
3	7.500	0.007	0.013	0.00	0.00	0.00	3	0.000	5.000	8.000
4	9.063	0.000	0.046	0.00	0.00	0.00	4	14.737	15.258	51.439
5	0.000	0.000	0.044	4.56	21.28	0.00	5	0.439	21.281	51.439
6	7.499	7.516	0.026	0.00	0.00	0.00	-	-	-	-

Nota. Resultados del análisis del Modelo 10 generados por el software SAP2000.

Tabla 79

Resultados del análisis estructural – Modelo 10 (Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.022	5.58	6.29	0.00	1	6.291	5.582	11.164
2	4.172	0.004	0.019	0.00	0.00	0.00	2	6.291	5.582	16.356
3	7.424	0.007	0.013	0.00	0.00	0.00	3	0.000	5.000	8.000
4	9.004	0.000	0.045	0.00	0.00	0.00	4	14.759	15.351	51.418
5	0.000	0.000	0.044	4.58	21.29	0.00	5	0.418	21.291	51.418
6	7.423	7.440	0.026	0.00	0.00	0.00	-	-	-	-

Nota. Resultados del análisis del Modelo 10 generados por el software Autodesk Robot Structural Analysis.

Tabla 80

Resultados del análisis estructural – Modelo 10 (OpenRSE)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.022	5.58	6.29	0.00	1	6.291	5.582	11.164
2	4.173	0.004	0.019	0.00	0.00	0.00	2	6.291	5.582	16.357
3	7.425	0.007	0.013	0.00	0.00	0.00	3	0.000	5.000	8.000
4	9.005	0.000	0.045	0.00	0.00	0.00	4	14.759	15.351	51.418
5	0.000	0.000	0.044	4.58	21.29	0.00	5	0.418	21.291	51.418
6	7.425	7.442	0.026	0.00	0.00	0.00	-	-	-	-

Nota. Resultados del análisis del Modelo 10 generados por el software OpenRSE, desarrollado en la presente investigación.

Para validar los resultados de OpenRSE, se presentan las diferencias absolutas, ya que permiten comparaciones directas sin distorsión en magnitudes pequeñas o grandes.

Tabla 81

Errores absolutos – Modelo 10 (OpenRSE vs. SAP2000)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.021	0.010	0.000	1	0.010	0.021	0.041
2	0.037	0.000	0.000	0.000	0.000	0.000	2	0.010	0.021	0.112
3	0.075	0.000	0.000	0.000	0.000	0.000	3	0.000	0.000	0.000
4	0.058	0.000	0.001	0.000	0.000	0.000	4	0.022	0.093	0.021
5	0.000	0.000	0.000	0.022	0.011	0.000	5	0.021	0.010	0.021
6	0.074	0.074	0.000	0.000	0.000	0.000	-	-	-	-

Nota. Errores absolutos calculados, basados en la comparación de los resultados de OpenRSE.

Tabla 82

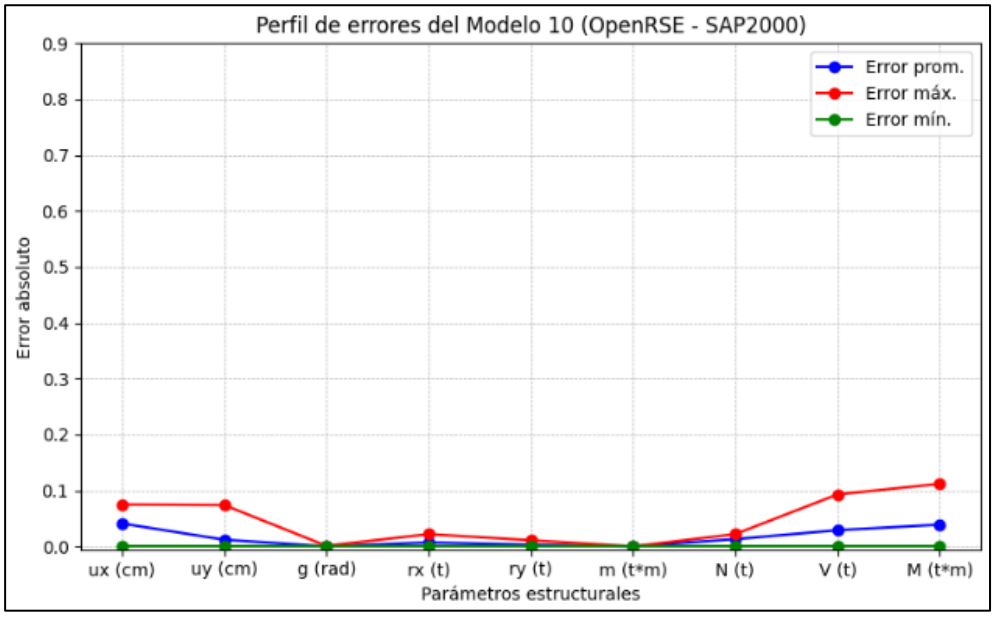
Resumen estadístico de errores absolutos – Modelo 10 (OpenRSE vs. SAP2000)

Medida	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.041	0.012	0.000	0.007	0.003	0.000	0.013	0.029	0.039
Err. max.	0.075	0.074	0.001	0.022	0.011	0.000	0.022	0.093	0.112
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Nota. El mayor error registrado corresponde al momento flector (0.112 t*m), seguido por la cortante (0.093 t), ambos dentro de márgenes aceptables para análisis estructurales, sin comprometer la precisión del modelo.

Figura 62

Perfil de errores absolutos – Modelo 10 (OpenRSE vs. SAP2000)



Nota. El gráfico evidencia que los errores máximos se concentran principalmente en los momentos flectores y desplazamientos, mientras que en la mayoría de los parámetros los errores mínimos son nulos y los errores promedio se mantienen bajos.

Tabla 83

Errores absolutos – Modelo 10 (OpenRSE vs. Robot Structural)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	b	N max.	V max.	M Max
1	0.000	0.000	0.000	0.000	0.000	0.000	1	0.000	0.000	0.000
2	0.001	0.000	0.000	0.000	0.000	0.000	2	0.000	0.000	0.001
3	0.001	0.000	0.000	0.000	0.000	0.000	3	0.000	0.000	0.000
4	0.001	0.000	0.000	0.000	0.000	0.000	4	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000	0.000	5	0.000	0.000	0.000
6	0.002	0.002	0.000	0.000	0.000	0.000	-	-	-	-

Nota. Errores absolutos calculado, basados en la comparación de los resultados de OpenRSE.

Tabla 84

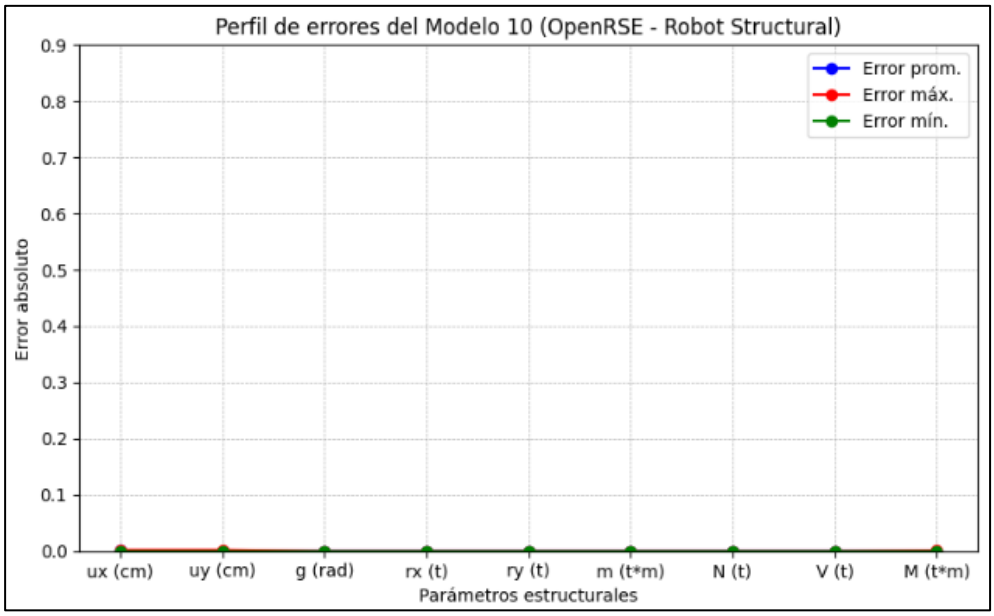
Resumen estadístico de errores absolutos – Modelo 10 (OpenRSE vs. Robot Structural)

Medida	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	m (t*m)	N max.	V max.	M Max
Err. prom.	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Err. max.	0.002	0.002	0.000	0.000	0.000	0.000	0.000	0.000	0.001
Err. min.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Nota. Los resultados evidencian una coincidencia prácticamente exacta, con errores absolutos nulos o despreciables en todos los parámetros estructurales evaluados.

Figura 63

Perfil de errores absolutos – Modelo 10 (OpenRSE vs. Robot Structural)



Nota. Todos los errores resultaron cercanos a cero, lo que refleja una alta precisión en los resultados obtenidos por OpenRSE.

Resumen de errores relativos porcentuales máximos

Resumen de errores relativos porcentuales máximos Con el objetivo de ofrecer una visión comparativa integral del comportamiento de OpenRSE frente a los programas de referencia, se presenta a continuación un resumen de los errores porcentuales relativos máximos registrados para cada parámetro estructural. Estos valores han sido obtenidos al comparar el error absoluto máximo con el valor máximo alcanzado por dicho parámetro, considerando todos los modelos analizados. La comparación se realiza tanto con SAP2000 como con Robot Structural, lo que permite dimensionar, en términos relativos, las diferencias más significativas observadas en cada caso. Esto contribuye a validar el nivel de precisión de OpenRSE en diversos escenarios de análisis estructural.

Tabla 85

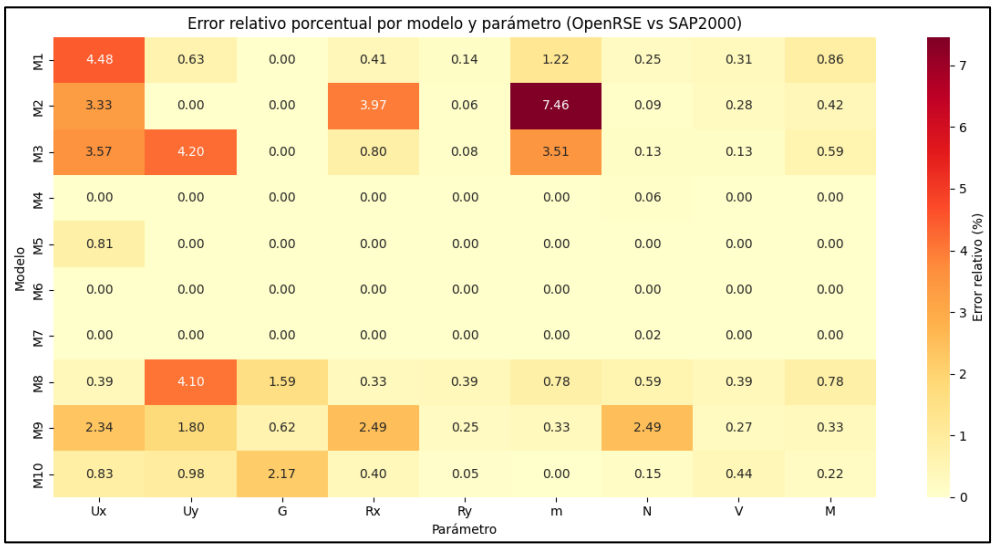
Errores relativos porcentuales máximos (OpenRSE vs SAP2000)

Mod.	Ux	Uy	G	Rx	Ry	m	N	V	M
1	4.48 %	0.63 %	0.00 %	0.41 %	0.14 %	1.22 %	0.25 %	0.31 %	0.86 %
2	3.33 %	0.00 %	0.00 %	3.97 %	0.06 %	7.46 %	0.09 %	0.28 %	0.42 %
3	3.57 %	4.20 %	0.00 %	0.80 %	0.08 %	3.51 %	0.13 %	0.13 %	0.59 %
4	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.06 %	0.00 %	0.00 %
5	0.81 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
6	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
7	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.02 %	0.00 %	0.00 %
8	0.39 %	4.10 %	1.59 %	0.33 %	0.39 %	0.78 %	0.59 %	0.39 %	0.78 %
9	2.34 %	1.80 %	0.62 %	2.49 %	0.25 %	0.33 %	2.49 %	0.27 %	0.33 %
10	0.83 %	0.98 %	2.17 %	0.40 %	0.05 %	0.00 %	0.15 %	0.44 %	0.22 %

Nota. Los valores indican el error relativo porcentual máximo entre OpenRSE y SAP2000, calculado como la razón entre el error absoluto máximo y el valor máximo del parámetro. Se observa alta concordancia entre ambos, destacando un máximo de 7.46 % en el momento de reacción del Modelo 2, mientras que la mayoría de los errores permanecen por debajo del 5 %.

Figura 64

Mapa de calor de errores relativos porcentuales máximos (OpenRSE vs SAP2000)



Nota. En la mayoría de los casos el error relativo es nulo; sin embargo, se observan porcentajes más elevados en los momentos de reacción y desplazamientos.

Tabla 86

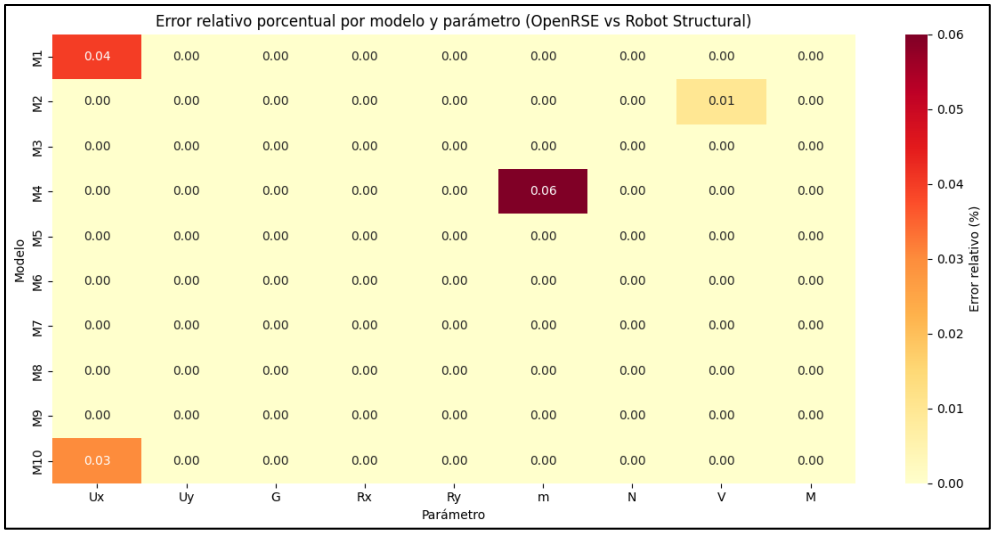
Errores relativos porcentuales máximos (OpenRSE vs Robot Structural)

Mod.	Ux	Uy	G	Rx	Ry	m	N	V	M
1	0.04 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
2	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.01 %	0.00 %
3	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
4	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.06 %	0.00 %	0.00 %	0.00 %
5	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
6	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
7	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
8	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
9	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
10	0.03 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %

Nota. Los valores muestran una coincidencia prácticamente exacta entre OpenRSE y Robot Structural, con errores relativos máximos inferiores al 0.1 % en todos los parámetros y modelos analizados.

Figura 65

Mapa de calor de errores relativos porcentuales máximos (OpenRSE vs Robot Structural)



Nota. El gráfico evidencia una coincidencia prácticamente exacta entre OpenRSE y Robot Structural, con errores relativos inferiores al 0.1 % en todos los modelos y parámetros evaluados.

4.1.4. PROCESAMIENTO COMPARATIVO DE EFICIENCIA OPERATIVA Y ACCESIBILIDAD DEL SOFTWARE

Con el propósito de ofrecer una evaluación integral del software desarrollado (OpenRSE) en comparación con herramientas comerciales de referencia, se han considerado criterios clave de eficiencia operativa y accesibilidad, enfocados en su impacto práctico para usuarios finales, especialmente en contextos educativos o de investigación.

Tiempo de procesamiento

Durante el análisis de los diez modelos estructurales, se pudo comprobar que el tiempo de cálculo, es decir, el tiempo requerido para obtener resultados una vez ingresado el modelo, resulta prácticamente indistinto entre OpenRSE, SAP2000 y Robot Structural. Este comportamiento se debe a que, a pesar de las diferencias en arquitectura y entorno, los algoritmos implementados por los tres programas son suficientemente optimizados y las capacidades computacionales actuales incluyendo dispositivos de cálculo como el TI-Nspire CX CAS, permiten resolver este tipo de análisis estructurales en tiempos insignificantes (menores a 1 segundo en todos los casos).

Es importante señalar que, si bien el tiempo de procesamiento es comparable, este se logra en contextos distintos de complejidad. Los programas comerciales poseen arquitecturas más pesadas, múltiples procesos de verificación internos y motores gráficos avanzados, lo cual justifica su similitud en tiempo respecto a un programa más liviano como OpenRSE. En síntesis, el tiempo de resolución no representa una ventaja ni desventaja diferencial entre los programas comparados.

Compatibilidad con sistemas operativos

La portabilidad del software es un aspecto crucial para su adopción. OpenRSE se ejecuta en una gama más amplia de plataformas, incluyendo:

- Windows
- macOS
- iPadOS (mediante TI-Nspire CX CAS App)
- Dispositivo TI-Nspire CX CAS físico

Por su parte, Robot Structural está disponible únicamente para Windows y macOS, mientras que SAP2000 es exclusivo para sistemas operativos Windows. Esto posiciona a OpenRSE como el más accesible en términos de compatibilidad.

Licenciamiento

En lo referente al acceso, tanto SAP2000 como Robot Structural operan bajo licencias propietarias de alto costo, que pueden superar los miles de dólares anualmente, restringiendo su disponibilidad a instituciones con presupuestos elevados o con acceso a licencias educativas. En contraste, OpenRSE es un programa de acceso libre, lo que lo hace especialmente útil en entornos académicos donde se busca reducir las barreras de entrada al software de análisis estructural.

Tamaño del archivo de instalación

Un criterio muchas veces subestimado pero importante es el tamaño del software. SAP2000 requiere un mínimo de 6 GB de espacio para su instalación, y Robot Structural alrededor de 5 GB. OpenRSE, en cambio, tiene un tamaño total de 25 KB, lo que lo convierte en un software extremadamente liviano y portable, fácilmente compartible o instalable incluso en dispositivos de bajo almacenamiento.

Tabla 87

Resumen comparativo de eficiencia operativa y accesibilidad entre programas analizados

Indicador	SAP2000	Robot Structural	OpenRSE
Tiempo de procesamiento	~1 segundo	~1 segundo	~1 segundo
Sistemas operativos compatibles	Windows	Windows, macOS	Windows, macOS, iPadOS, TI-Nspire
Tipo de licencia	Privativo	Privativo	Libre
Tamaño de instalación	6 GB	5 GB	25KB

Nota. OpenRSE presenta ventajas significativas en términos de accesibilidad, portabilidad y simplicidad operativa, sin comprometer los tiempos de procesamiento respecto a los softwares comerciales.

4.2. CONTRASTACIÓN DE HIPÓTESIS Y PRUEBA DE HIPÓTESIS

H1: Si se identifican e incorporan las características adecuadas en el desarrollo de OpenRSE en Lua, entonces se mejorará la eficiencia en el análisis de estructuras hiperestáticas en Huánuco, 2024.

El análisis del desarrollo de OpenRSE, especialmente lo expuesto en el apartado 4.1.1, demuestra que la adopción de una estructura modular orientada a objetos fue una decisión clave para mejorar la eficiencia operativa del programa. Este enfoque permite que las funciones se ejecuten de forma distribuida y no secuencial, lo cual reduce considerablemente la sobrecarga computacional. Como resultado, OpenRSE logra tiempos de respuesta y ejecución que se sitúan al nivel de programas comerciales como SAP2000 y Robot Structural, incluso al ejecutarse en dispositivos de capacidad limitada como la calculadora TI-Nspire CX CAS.

Otra característica destacable es la interfaz gráfica de usuario, compuesta por 21 ventanas específicas que guían al usuario en todo el flujo de análisis estructural, desde el ingreso de datos hasta la visualización de

resultados. Esta interfaz ha sido diseñada con enfoque en la simplicidad, claridad y eficiencia de uso. Un ejemplo de ello es la asignación de restricciones, OpenRSE permite seleccionar apoyos estructurales a partir de ocho configuraciones posibles de grados de libertad, incluyendo control de rotación mediante una selección directa y rápida. Esta facilidad en la definición de condiciones de contorno resalta tanto la accesibilidad como la eficiencia operativa del programa.

Una funcionalidad particularmente innovadora es la visualización completa del proceso de cálculo: matrices, vectores y cada etapa del análisis estructural se presentan al usuario, lo que permite una verificación detallada y comprensible del procedimiento. A diferencia de los programas comerciales, donde los resultados se presentan como una caja negra, OpenRSE ofrece transparencia total, facilitando su uso en contextos académicos y profesionales.

Además, el sistema permite verificar resultados con alta precisión decimal en puntos específicos, sin limitarse a discretizaciones gráficas, lo que incrementa aún más su utilidad práctica. Sumado a su peso de apenas 25 KB, todas estas características demuestran que OpenRSE es un software altamente eficiente, no solo en su lógica de cálculo, sino también en su diseño visual, velocidad de ejecución y facilidad de operación.

H2: Si se integran metodologías avanzadas de análisis estructural bidimensional de estructuras hiperestáticas en OpenRSE, entonces se garantizará un análisis más preciso y eficiente en Huánuco, 2024.

OpenRSE integra una metodología robusta y avanzada para el análisis estructural bidimensional, el método matricial. Este enfoque es ampliamente utilizado en ingeniería estructural por su capacidad de resolver sistemas hiperestáticos con precisión y adaptabilidad. En OpenRSE, esta metodología ha sido implementada con alto nivel de detalle dentro de la clase analizar, compuesta por más de 680 líneas de código y segmentada en funciones específicas. Cada etapa del cálculo, desde la obtención de longitudes y ángulos de barra, la generación de matrices locales y globales, hasta la

solución del sistema de ecuaciones ha sido automatizada y organizada para asegurar claridad, eficiencia y trazabilidad (ver apartado 4.1.2).

Un punto clave de esta implementación es el procesamiento automatizado de los grados de libertad. A través del método clasificar_gdl(), el programa interpreta las restricciones estructurales a partir del tipo de apoyo seleccionado por el usuario, sin necesidad de intervención avanzada. Este sistema contempla ocho configuraciones posibles, incluyendo control de rotación, y permite definir de manera clara y directa las condiciones de borde. En contraste, los programas comerciales especializados suelen presentar esta funcionalidad de forma poco explícita, lo que puede generar confusión en el usuario, ya que la representación de restricciones no siempre refleja con claridad el tipo exacto de condición impuesta.

Además, el sistema de clasificación inteligente de grados de libertad incorpora una lógica adicional: si un nodo está conectado exclusivamente mediante articulaciones, el programa restringe automáticamente su rotación, evitando errores de interpretación estructural sin intervención manual. Este tipo de automatización refuerza la eficiencia del análisis y reduce significativamente el margen de error por parte del usuario.

En conjunto, la integración de esta metodología avanzada, junto con su implementación clara, modular y automatizada, permite a OpenRSE ofrecer un análisis preciso y eficiente. El programa no solo reproduce los cálculos fundamentales con fidelidad, sino que mejora la experiencia del usuario al facilitar tareas que en otros entornos requieren conocimientos técnicos más avanzados.

H3: Si se optimiza OpenRSE en Lua mediante mejoras en su código y algoritmos, entonces se incrementará la precisión y la velocidad en la verificación de estructuras hiperestáticas bidimensionales en Huánuco, 2024.

La optimización de código fue uno de los enfoques centrales del desarrollo. La eficiencia algorítmica se ve reflejada en la estructura clara y

modular del código, con más de 3500 líneas organizadas en funciones específicas, evitando procesos redundantes o procedimientos costosos en recursos.

Respecto a la precisión, se evaluaron 10 modelos estructurales de diversas tipologías. En todos los casos, los errores absolutos entre OpenRSE y los programas comerciales fueron mínimos. Por ejemplo, el error absoluto máximo en desplazamientos (ux) frente a SAP2000 fue de 0.127 cm, mientras que frente a Robot Structural fue de apenas 0.001 cm. Asimismo, los errores relativos porcentuales máximos rara vez superaron el 5 %, siendo mayormente inferiores al 1 %, y en muchos casos cero.

Además, el tiempo de cálculo en los tres programas fue prácticamente indistinto (~1 segundo), lo que refleja que la optimización del código de OpenRSE permite una verificación ágil y comparable a softwares consolidados.

H4: Si se validan los resultados de OpenRSE comparándolos con los obtenidos por SAP2000 y Robot Structural, entonces se confirmará su robustez y precisión en el análisis de estructuras hiperestáticas en Huánuco, 2024.

La validación se realizó mediante un análisis comparativo de los resultados obtenidos por los tres programas, aplicando tanto errores absolutos como errores relativos porcentuales. El estudio abarcó desplazamientos, reacciones y esfuerzos internos (N, V, M), considerando tanto valores máximos por parámetro como distribuciones completas.

Los resultados fueron contundentes, las diferencias con Robot Structural fueron prácticamente nulas (errores < 0.1 %), mientras que con SAP2000 las discrepancias fueron leves y atribuibles a diferencias internas en los modelos numéricos o criterios de redondeo. En ningún caso los errores superaron márgenes aceptables para ingeniería estructural. Por ejemplo, en el parámetro con mayor diferencia (momento en el Modelo 2), el error relativo

con SAP2000 fue de 7.46 %, dentro de los márgenes esperables por variaciones en discretización y criterios de cálculo.

Esta validación cruzada con dos de los programas más utilizados del mercado refuerza la confiabilidad y robustez del motor de análisis de OpenRSE.

H5: Si se evalúa el desempeño de OpenRSE en Lua en comparación con SAP2000 y Robot Structural, entonces se demostrará que contribuye en mayor medida a la eficiencia y accesibilidad para los ingenieros en Huánuco, 2024.

Los indicadores de eficiencia y accesibilidad fueron abordados integralmente. En cuanto a eficiencia operativa, OpenRSE iguala a los programas comerciales en tiempos de procesamiento, incluso en entornos con recursos limitados como el TI-Nspire CX CAS.

En términos de accesibilidad, OpenRSE supera ampliamente a sus pares. Es de licencia libre, frente a los altos costos de SAP2000 y Robot Structural; su archivo de instalación es de 25 KB, comparado con los 5 - 6 GB de los programas comerciales; y es compatible con una gama más amplia de sistemas operativos, incluyendo Windows, macOS, iPadOS y dispositivos portátiles. Esto facilita su implementación en entornos educativos, zonas con limitaciones tecnológicas o proyectos de bajo presupuesto.

HG: Si se desarrolla OpenRSE en Lua, entonces mejorará la eficiencia y accesibilidad en el análisis estructural bidimensional de estructuras hiperestáticas en Huánuco, 2024.

La validación conjunta de todas las hipótesis específicas permite afirmar que la hipótesis general se cumple. El desarrollo de OpenRSE ha permitido mejorar tanto la eficiencia operativa (flujo lógico de análisis, tiempos reducidos, precisión validada) como la accesibilidad (licencia libre, portabilidad multiplataforma, tamaño reducido). Además, su estructura abierta

y transparente representa una ventaja formativa frente a soluciones de tipo caja negra.

La validación por comparación con SAP2000 y Robot Structural ha demostrado que el programa mantiene altos estándares de calidad técnica, y su diseño lo convierte en una herramienta de gran utilidad práctica y académica para el análisis de estructuras hiperestáticas.

CAPÍTULO V
DISCUSIÓN DE RESULTADOS

Los resultados obtenidos a lo largo de esta investigación permiten interpretar de forma clara la relación entre el desarrollo de OpenRSE y su impacto en la eficiencia y accesibilidad del análisis estructural bidimensional de estructuras hiperestáticas. Para poder concluir con certeza que el programa contribuye efectivamente a estos dos factores, primero fue esencial validar la precisión numérica de los resultados generados. Esta validación fue el eje central del análisis, ya que sin ella no sería posible afirmar que las mejoras propuestas realmente cumplen con su propósito.

Se plantearon un conjunto de diez modelos estructurales variados, seleccionados estratégicamente para representar distintas configuraciones reales en cuanto a geometría, materiales, condiciones de contorno y tipo estructural. Dado que no existe una población definida de estructuras posibles y que las combinaciones son prácticamente infinitas, la elección de modelos se orientó a cubrir un amplio rango de situaciones comunes en el análisis estructural, manteniendo el enfoque bidimensional propuesto por la investigación. Cada uno de estos modelos fue resuelto utilizando SAP2000, Robot Structural y OpenRSE, lo que permitió hacer comparaciones directas entre los resultados obtenidos.

La diferencia entre los programas se cuantificó mediante errores absolutos entre los resultados de OpenRSE y los de los programas de referencia. En una etapa posterior, se establecieron los errores relativos porcentuales máximos, calculados como la relación entre el error absoluto máximo y el valor máximo del parámetro correspondiente. Este análisis, detallado en el apartado 4.1.3 y resumido en las tablas 85 y 86, permitió observar que la mayor diferencia porcentual fue de 7.46 %, mientras que la mayoría de los resultados mostraron errores cercanos al 0 %, especialmente en la comparación con Robot Structural. En este caso, los errores relativos fueron prácticamente nulos en todos los modelos y parámetros, lo que indica

una coincidencia total en los resultados obtenidos por ambos programas. Esto no solo valida la precisión de OpenRSE, sino que también permite suponer que sigue un paradigma de cálculo muy similar al de Robot Structural.

En cuanto a SAP2000, los errores encontrados fueron igualmente bajos, aunque ligeramente más altos que con Robot Structural. Esto no debe interpretarse como una falla en el programa desarrollado, sino como una diferencia natural derivada de los procesos internos de cálculo de SAP2000. Este software, al ser más robusto y cerrado en su funcionamiento, puede incluir factores de corrección, redondeo o seguridad que no están explícitamente documentados para el usuario, lo que puede explicar algunas variaciones en los resultados. OpenRSE, al contrario, presenta un modelo de cálculo completamente transparente, lo que permite identificar claramente cada paso del análisis, revisar cada matriz o vector involucrado y verificar el desarrollo de manera detallada.

Estos resultados aportan a la comprensión del problema de investigación, ya que demuestran que es posible desarrollar un software libre, liviano y transparente, capaz de generar resultados precisos en el análisis estructural bidimensional. Esta comprobación es clave para validar las hipótesis propuestas, las cuales apuntan a la mejora de la eficiencia y la accesibilidad en este tipo de análisis.

Desde el punto de vista metodológico, esta investigación se sitúa dentro de un enfoque cuantitativo y explicativo. El diseño experimental aplicado permitió tomar teorías y métodos existentes del análisis estructural, principalmente los desarrollados por Quispe Panca (2015), y llevarlos a un contexto computacional utilizando el lenguaje Lua sobre la plataforma TI-Nspire. A diferencia de estudios previos que presentaban el cálculo estructural como un conjunto de situaciones aisladas, esta investigación los integró en un único sistema funcional y coherente, demostrando su aplicabilidad a través de programación estructurada.

El desarrollo del programa y su validación también tienen implicaciones prácticas importantes. Se demuestra que es posible lograr niveles de precisión comparables a los de software comercial utilizando herramientas de bajo costo o incluso gratuitas. Esto representa una contribución significativa en contextos como el nacional, donde el acceso a programas licenciados puede ser limitado por razones económicas. Muchos profesionales se ven obligados a utilizar versiones modificas de software, lo cual, además de ser ilegal, implica riesgos de seguridad informática. OpenRSE se presenta como una alternativa viable, legal y segura, con la capacidad de ser utilizada tanto en educación como en investigación o aplicación profesional.

Entre las fortalezas del proyecto destaca la arquitectura modular del programa, que no solo favorece la eficiencia y la transparencia, sino que permite futuras extensiones o mejoras sin alterar la base existente. En cuanto a las limitaciones, es evidente que el análisis se restringe al ámbito bidimensional y que no se han implementado funcionalidades avanzadas como análisis dinámicos, cargas complejas o modelado tridimensional. También se reconoce que el método de análisis utilizado, si bien es altamente eficiente, puede ser superado en algunos aspectos por métodos más complejos como el análisis por elementos finitos. No obstante, el desarrollo de estas funcionalidades requiere de un equipo de trabajo especializado y mayor disponibilidad de tiempo, lo cual excede el alcance planteado en esta investigación.

En síntesis, esta investigación demuestra que un desarrollo independiente, con recursos limitados y fundamentado en teorías sólidas, puede lograr resultados válidos, precisos y útiles en el campo del análisis estructural. OpenRSE no solo cumple con los objetivos planteados, sino que abre la posibilidad de continuar construyendo herramientas accesibles y eficientes para la ingeniería estructural.

CONCLUSIONES

Objetivo: Determinar de qué manera influye el desarrollo de OpenRSE en Lua en la mejora de la eficiencia y accesibilidad en el análisis estructural bidimensional de estructuras hiperestáticas, Huánuco, 2024.

Los resultados obtenidos en esta investigación permiten concluir que el desarrollo de OpenRSE, bajo lenguaje Lua y ejecutado sobre la plataforma TI-Nspire CX CAS, ha alcanzado un desempeño técnico destacado en el análisis estructural bidimensional de estructuras hiperestáticas, tanto por su eficiencia operativa como por su alto nivel de accesibilidad. El sistema, compuesto por un total de 3541 líneas de código distribuidas modularmente, integra una arquitectura orientada a objetos que permite una ejecución rápida, estable y trazable, replicando con éxito las etapas del análisis estructural profesional mediante métodos matriciales. Esta organización ha demostrado ser funcional en diversos entornos, desde estaciones de trabajo hasta dispositivos portátiles, lo que refuerza su accesibilidad para estudiantes, investigadores o profesionales sin acceso a equipos avanzados.

Objetivo 1: Identificar las características que debe incluir el desarrollo de OpenRSE en Lua para mejorar la eficiencia y accesibilidad en el análisis de estructuras hiperestáticas, Huánuco, 2024.

Se identificaron como características clave que OpenRSE permite modelar cualquier sistema estructural bidimensional que pueda resolverse mediante el método matricial, sin que el usuario deba especificar previamente la naturaleza del sistema (como viga, pórtico o armadura). Esta capacidad mejora sustancialmente la eficiencia operativa, al eliminar pasos intermedios de configuración, y eleva la accesibilidad técnica al simplificar el proceso de modelado. Cada barra se interpreta automáticamente según sus condiciones de frontera, lo que permite representar una amplia gama de casos estructurales con mínima intervención. Adicionalmente, la validación funcional de OpenRSE no solo se evidencia en la precisión de sus resultados, sino

también en la claridad con la que se presentan. A diferencia de los programas comerciales, OpenRSE ofrece al usuario la posibilidad de visualizar matrices, vectores y procesos intermedios. Esta característica mejora tanto la eficiencia del aprendizaje como la comprensión estructural del modelo, siendo una herramienta de gran valor en entornos académicos o formativos.

Objetivo 2: Integrar metodologías de análisis estructural bidimensional de estructuras hiperestáticas en OpenRSE para garantizar un análisis preciso y eficiente, Huánuco, 2024.

Se integraron exitosamente metodologías de análisis basadas en el método matricial, como parte de una arquitectura orientada a objetos. Esta integración es la base que permite la ejecución rápida, estable y trazable del análisis, garantizando así la precisión y eficiencia de los resultados, los cuales fueron posteriormente confirmados mediante validación contra software de referencia.

Objetivo 3: Optimizar OpenRSE en Lua para mejorar la precisión y la velocidad en la verificación de estructuras hiperestáticas bidimensionales, Huánuco, 2024.

En términos de eficiencia computacional, el software ejecuta cálculos completos en menos de un segundo, equiparándose en tiempo de respuesta a los programas comerciales SAP2000 y Robot Structural. Esta rapidez se alcanza gracias al diseño optimizado del código y a la estructura modular del programa, que minimiza la carga de procesamiento sin sacrificar la precisión. Un aspecto particularmente destacable es que OpenRSE logra este rendimiento incluso en plataformas de bajo consumo como la calculadora TI-Nspire CX CAS, lo que demuestra su eficiencia incluso fuera de entornos convencionales.

Objetivo 4: Validar los resultados de OpenRSE comparándolos con los obtenidos por SAP2000 y Robot Structural, para garantizar su robustez y precisión en el análisis de estructuras hiperestáticas, Huánuco, 2024.

La comparación sistemática con programas de referencia como SAP2000 y Robot Structural demuestra una alta precisión numérica en el análisis estructural bidimensional de modelos hiperestáticos. El análisis de diez modelos estructurales con distintas configuraciones permitió obtener errores absolutos mínimos, siendo el mayor error relativo porcentual de 7.46 % únicamente en un caso específico, mientras que en la mayoría de los parámetros y modelos se observaron diferencias por debajo del 1 %, o incluso del 0 %. En el caso de Robot Structural, la coincidencia con los resultados de OpenRSE fue prácticamente exacta, con diferencias que no superan el 0.1 %. Esta precisión confirma que OpenRSE no solo es técnicamente válido, sino que puede ser utilizado con confianza para resolver problemas reales de análisis estructural.

Objetivo 5: Evaluar en qué medida el desarrollo de OpenRSE en Lua contribuye a mejorar la eficiencia y accesibilidad para los ingenieros en comparación con SAP2000 y Robot Structural, Huánuco, 2024.

En lo que respecta a la accesibilidad tecnológica, OpenRSE supera ampliamente a sus pares comerciales. Su compatibilidad con múltiples sistemas operativos (Windows, macOS, iPadOS y el entorno TI-Nspire) contrasta con la limitada disponibilidad de SAP2000 y Robot Structural. Además, el archivo de instalación de OpenRSE ocupa solo 25 KB, frente a los más de 5 GB requeridos por los programas comerciales, lo cual refuerza su portabilidad y facilidad de distribución. Su licencia libre también elimina barreras de acceso, permitiendo que el software pueda ser utilizado sin restricciones económicas, una ventaja fundamental para instituciones educativas, profesionales independientes o estudiantes.

RECOMENDACIONES

En vista de estos resultados, se recomienda continuar el desarrollo del software OpenRSE ampliando su alcance hacia el análisis tridimensional de estructuras, lo cual permitiría abarcar una gama más amplia de aplicaciones prácticas. Para ello, se sugiere conformar un equipo de trabajo multidisciplinario que integre especialistas en análisis estructural, programación avanzada y diseño de interfaces, con el fin de mantener la eficiencia operativa del sistema sin perder su simplicidad y accesibilidad. La base modular actual permite estas extensiones de manera progresiva sin necesidad de reestructurar el núcleo del sistema.

Asimismo, sería conveniente implementar un módulo de análisis dinámico, incluyendo cargas sísmicas, análisis espectral y respuesta en el tiempo. Esto permitiría responder a requisitos normativos actuales y ampliar las capacidades del software en contextos de diseño estructural más complejos. Se sugiere también el desarrollo de herramientas de discretización automatizada para cargas variables o distribuidas no uniformemente, lo cual incrementaría la precisión de análisis en casos reales más detallados.

Se recomienda mantener la filosofía de acceso libre y multiplataforma, ya que estos aspectos representan una ventaja competitiva frente a soluciones comerciales, particularmente en contextos académicos, zonas rurales o instituciones con recursos limitados. Igualmente, se aconseja la documentación técnica extensa y la elaboración de guías de usuario didácticas para facilitar el aprendizaje y la adopción por parte de nuevos usuarios.

Finalmente, se considera relevante la difusión de esta herramienta en espacios de formación profesional y académica, como universidades e institutos técnicos, ya que OpenRSE representa una alternativa viable, funcional y educativa frente a las soluciones comerciales, y puede contribuir

significativamente al fortalecimiento del conocimiento en análisis estructural sin depender de recursos económicos elevados o licencias privativas.

REFERENCIAS BIBLIOGRÁFICAS

Aguiar Falconi, R. (2004). *Análisis matricial de estructuras* (3ª ed.). CEINCI-ESPE.

Asmal Rodas, P. M., & Yumbla Cadme, L. O. (2023). *Implementación y validación de un programa basado en el método de elementos finitos para la solución de problemas cuasiestáticos en cuerpos bidimensionales* [Tesis de pregrado, Universidad de Cuenca]. http://dspace.ucuenca.edu.ec/handle/123456789/42035

Bazán, E., & Meli, R. (2002). *Diseño sísmico de edificios*. Limusa.

Blanco Claraco, J. L., González Herrera, A., & García-Manrique Ocaña, J. M. (2012). *Análisis estático de estructuras por el método matricial*. Universidad de Málaga.

Celigueta Lizarza, J. T. (1998). *Curso de análisis estructural*. EUNSA.

Celigueta Lizarza, J. T. (2008). *Método de los elementos finitos para análisis estructural*. Tecnun.

Cervera, M., & Blanco, E. (2004). *Mecánica de estructuras* (2ª ed.). Edicions UPC.

Chandrupatla, T. R., & Belegundu, A. D. (1999). *Introducción al estudio del elemento finito en ingeniería* (2ª ed.). Pearson.

Comexperu. (2024). *Salario mínimo y productividad laboral.* https://www.comexperu.org.pe/

Construsoft. (2024). *Software de análisis y diseño estructural.* https://www.construsoft.es/

CSI Computers & Structures, Inc. (2024, 15 de julio). *CSI product pricing*. https://www.csiamerica.com/sales

Damy Ríos, J. (1986). *Apuntes del curso: Aplicación de las computadoras al análisis estructural. Facultad de Ingeniería*, UNAM.

De León León, J. D. (2020). *Creación de software para la línea estructural de ingeniería civil* [Tesis de pregrado, Universidad Cooperativa de Colombia]. https://hdl.handle.net/20.500.12494/35481

Díaz Barrantes, F. D., & Guillén Hernández, Á. R. (2020). *Modelo computacional para el análisis matricial de estructuras reticulares* [Tesis de pregrado, Universidad Peruana de Ciencias Aplicadas]. http://hdl.handle.net/10757/648845

El Blog de Python. (2024). *Comparativa: Python vs Lua: similitudes y diferencias en este blog de programación*. https://elblogpython.com/

González Cárceles, J. (1990). *Análisis del proceso de diseño de estructuras porticadas* [Tesis doctoral, Universidad Politécnica de Madrid].

Guerra Utrilla, G. C. (2022). *Análisis y diseño estructural con el software ETABS de un edificio comercial de 5 niveles de concreto armado, Huánuco - Huánuco - 2022* [Tesis de pregrado, Universidad de Huánuco]. http://repositorio.udh.edu.pe/20.500.14257/4007

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). *Metodología de la investigación* (6ª ed.). McGraw-Hill.

Hostman. (2024). *Lua vs Python: A comparison of performance and use cases*. https://hostman.com/

Incober S.L. (2024). *Análisis de estructuras*. https://www.incober.es/

Ierusalimschy, R., Figueiredo, L. H., & Celes, W. (2008). *Manual de referencia de Lua 5.1*. https://www.lua.org/manual/5.1/es/

Inza Ramírez, D. E., & Nación Ramos, E. E. (2024). *Diseño estructural de la superestructura y subestructura de un puente tipo viga-losa ubicado en el centro poblado de Jancao Bajo* [Tesis de pregrado, Universidad Nacional Hermilio Valdizán]. https://hdl.handle.net/20.500.13080/10139

López Pajuelo, J. G. (2022). *Propuesta de diseño estructural de un reservorio aplicando el software SAP2000 para el servicio de agua potable del centro poblado de Matibamba, Amarilis - 2022* [Tesis de pregrado, Universidad de Huánuco]. http://repositorio.udh.edu.pe/123456789/3835

Lupaca Quispe, D. R. (2022). *Desarrollo del simulador web Beli para el análisis matricial de estructuras planas y espaciales* [Tesis de pregrado, Universidad César Vallejo]. https://hdl.handle.net/20.500.12692/90939

Luthe, R. G. (s. f.). *Análisis estructural*. Alfaomega.

Martín Chica, F. (2018). *Determinación directa de la matriz reducida en el método de los desplazamientos*.

Mioti. (2024). *Python, el lenguaje de programación más popular en data science*. https://mioti.es/

Montúfar Chata, E. F. (2022*). Análisis comparativo del modelamiento y diseño estructural en concreto armado utilizando los softwares SAP2000, ETABS, CYPECAD y Revit Structure para la infraestructura educativa Sorapa* [Tesis de pregrado, Universidad Nacional del Altiplano]. http://repositorio.unap.edu.pe/handle/20.500.14082/18907

Muñoz, L. E. (2023). *Programación en Lua* (1ª ed.). CIMTED.

Pino Herrera, J., Martínez Moreno, P., Vergara Camacho, J. A., & Contreras Vega, G. (2020). *Introducción a la programación*. HESS.

Quispe Panca, A. J. (2015). *Análisis matricial de estructuras: Introducción al método de elementos finitos*. Macro EIRL.

Ramírez Vargas, C. D. (2022). *Desarrollo de un programa de computador para el análisis lineal de estructuras aporticadas tridimensionales sometidas a cargas estáticas* [Tesis de pregrado, Universidad Nacional de Colombia]. https://repositorio.unal.edu.co/handle/unal/81538

Rojas Rojas, R. M., & Padilla Punzo, H. M. (2009). *Análisis estructural con matrices*. Trillas.

Structuralia. (2021). *5 software utilizados para el diseño y cálculo de estructuras en edificación y obra civil*. https://blog.structuralia.com/

Tena Colunga, A. (2007). *Análisis de estructuras con métodos matriciales*. Limusa.

Texas Instruments. (s. f.). *Lua Scripting API Reference Guide*.

Vázquez Fernández, M. (1999). *Cálculo matricial de estructuras* (2ª ed.). Colegio de Ingenieros Técnicos de Obras Públicas de Madrid.

COMO CITAR ESTE TRABAJO DE INVESTIGACIÓN

Rosas Placido, A. (2025). *Desarrollo de OpenRSE en Lua para mejorar la eficiencia y accesibilidad en el análisis estructural bidimensional de estructuras hiperestáticas, Huánuco, 2024.* [Tesis de pregrado, Universidad de Huánuco]. Repositorio Institucional de la UDH. http://...

ANEXOS

ANEXO 1

MATRIZ DE CONSISTENCIA

"DESARROLLO DE OPENRSE EN LUA PARA MEJORAR LA EFICIENCIA Y ACCESIBILIDAD EN EL ANÁLISIS ESTRUCTURAL BIDIMENSIONAL DE ESTRUCTURAS HIPERESTÁTICAS, HUÁNUCO, 2024."

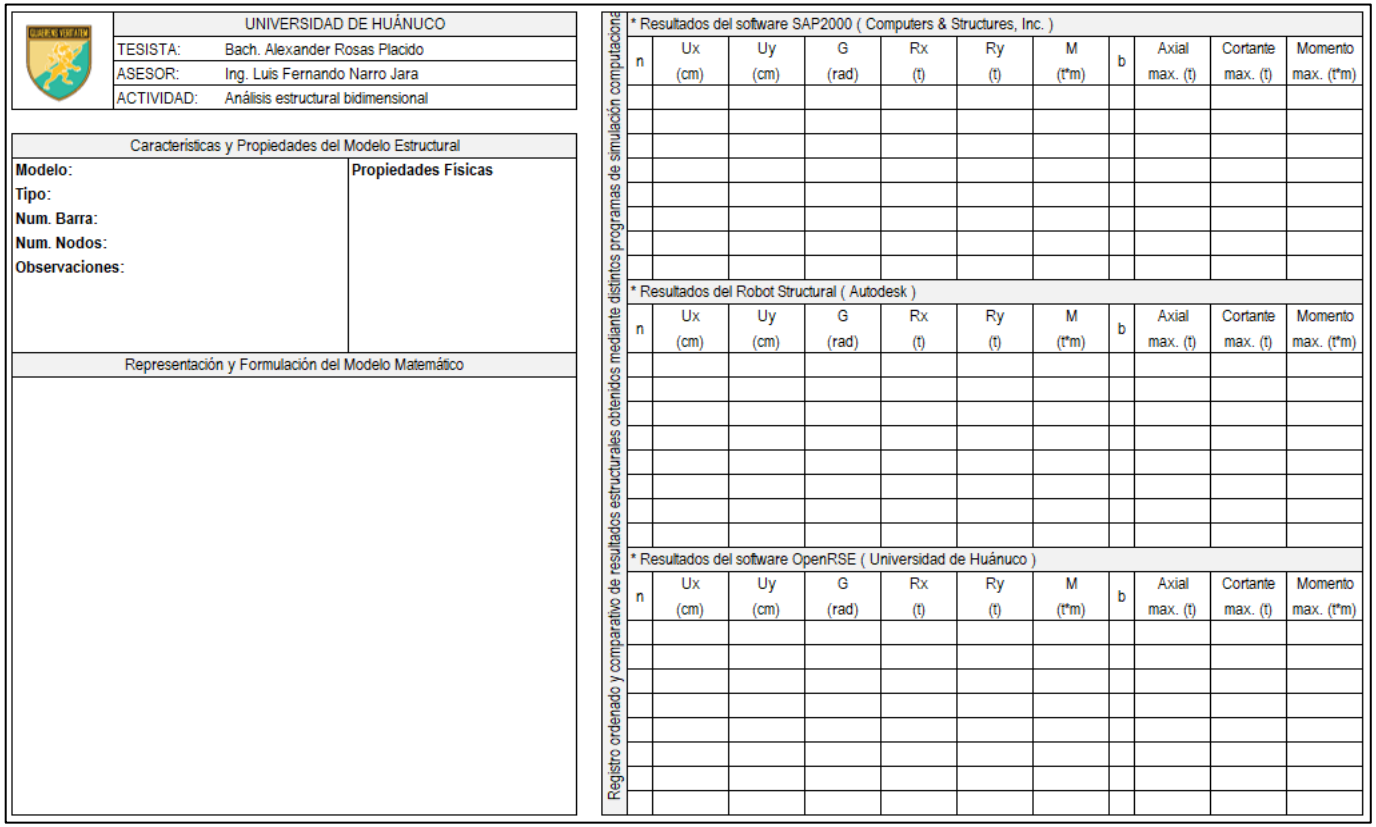
PROBLEMA	OBJETIVO	HIPÓTESIS	VARIABLE	METODOLOGÍA
Problema General: ¿De qué manera influye, en el análisis estructural bidimensional de estructuras hiperestáticas, el desarrollo de OpenRSE en Lua en la mejora de la eficiencia y accesibilidad, Huánuco, 2024? **Problemas Específicos:** 1. ¿Qué características debe incluir el desarrollo de OpenRSE en Lua para mejorar la eficiencia y eficiencia en el análisis de estructuras hiperestáticas, Huánuco, 2024? 2. ¿Qué metodologías de análisis estructural bidimensional de estructuras hiperestáticas deben integrarse en OpenRSE para garantizar un análisis preciso y eficiente, Huánuco, 2024? 3. ¿Cómo se puede optimizar OpenRSE en Lua para mejorar la precisión y la velocidad en la	**Objetivo General:** Determinar de qué manera influye el desarrollo de OpenRSE en Lua en la mejora de la eficiencia y accesibilidad en el análisis estructural bidimensional de estructuras hiperestáticas, Huánuco, 2024. **Objetivos Específicos:** 1. Identificar las características que debe incluir el desarrollo de OpenRSE en Lua para mejorar la eficiencia y eficiencia en el análisis de estructuras hiperestáticas, Huánuco, 2024. 2. Integrar metodologías de análisis estructural bidimensional de estructuras hiperestáticas en OpenRSE para garantizar un análisis preciso y eficiente, Huánuco, 2024.	**Hipótesis General:** Si se desarrolla OpenRSE en Lua, entonces mejorará la eficiencia y accesibilidad en el análisis estructural bidimensional de estructuras hiperestáticas en Huánuco, 2024. **Hipótesis Específicas:** 1. Si se identifican e incorporan las características adecuadas en el desarrollo de OpenRSE en Lua, entonces se mejorará la eficiencia en el análisis de estructuras hiperestáticas en Huánuco, 2024. 2. Si se integran metodologías avanzadas de análisis estructural bidimensional de estructuras hiperestáticas en OpenRSE, entonces se garantizará un análisis más preciso y eficiente en Huánuco, 2024. 3. Si se optimiza OpenRSE en Lua mediante mejoras en su código y algoritmos, entonces se incrementará la precisión y la	**Variable Independiente:** Desarrollo de OpenRSE en Lua **Variable Dependiente:** Eficiencia y accesibilidad en el análisis estructural bidimensional	**Tipo de Investigación** Cuantitativo **Nivel de investigación:** Explicativo **Diseño de Investigación:** Experimental **Población:** Modelos matemáticos bidimensional de estructuras hiperestáticas **Muestra:** No probabilístico o dirigido: modelos específicos **Técnica:** • Observación sistemática • Pruebas estandarizadas **Instrumentos:** • Metodológicos: Formato Excel de registro preciso y consistente de resultados obtenidos • Físicos: Computadora portátil, software TI-Nspire, dispositivo Handheld TI-Nspire, programas

PROBLEMA	OBJETIVO	HIPÓTESIS	VARIABLE	METODOLOGÍA
verificación de estructuras hiperestáticas bidimensionales, Huánuco, 2024? 4. ¿Cómo validar los resultados de OpenRSE comparándolos con los obtenidos por SAP2000 y Robot Structural, para garantizar su robustez y precisión en el análisis de estructuras hiperestáticas, Huánuco, 2024? 5. ¿En qué medida el desarrollo de OpenRSE en Lua contribuye a mejorar la eficiencia y accesibilidad para los ingenieros en comparación con SAP2000 y Robot Structural, Huánuco, 2024?	3. Optimizar OpenRSE en Lua para mejorar la precisión y la velocidad en la verificación de estructuras hiperestáticas bidimensionales, Huánuco, 2024. 4. Validar los resultados de OpenRSE comparándolos con los obtenidos por SAP2000 y Robot Structural, para garantizar su robustez y precisión en el análisis de estructuras hiperestáticas, Huánuco, 2024. 5. Evaluar en qué medida el desarrollo de OpenRSE en Lua contribuye a mejorar la eficiencia y accesibilidad para los ingenieros en comparación con SAP2000 y Robot Structural, Huánuco, 2024.	velocidad en la verificación de estructuras hiperestáticas bidimensionales en Huánuco, 2024. 4. Si se validan los resultados de OpenRSE comparándolos con los obtenidos por SAP2000 y Robot Structural, entonces se confirmará su robustez y precisión en el análisis de estructuras hiperestáticas en Huánuco, 2024. 5. Si se evalúa el desempeño de OpenRSE en Lua en comparación con SAP2000 y Robot Structural, entonces se demostrará que contribuye en mayor medida a la eficiencia y accesibilidad para los ingenieros en Huánuco, 2024.		especializados (SAP2000 y Robot Structural Analysis) **Técnica de procesamiento de datos:** Los parámetros de OpenRSE en LUA, SAP2000 y Robot Structural se documentarán en un formato Excel. Se analizará la conformidad y se registrarán los pasos del desarrollo. Los resultados se interpretarán y se presentarán con gráficos y tablas.

Nota. La matriz resume el diseño metodológico y la coherencia lógica de la presente investigación.

ANEXO 2

INSTRUMENTO DE RECOLECCIÓN DE DATOS



UNIVERSIDAD DE HUÁNUCO	
TESISTA:	Bach. Alexander Rosas Placido
ASESOR:	Ing. Luis Fernando Narro Jara
ACTIVIDAD:	Análisis estructural bidimensional

Características y Propiedades del Modelo Estructural	
Modelo:	**Propiedades Físicas**
Tipo:	
Num. Barra:	
Num. Nodos:	
Observaciones:	

Representación y Formulación del Modelo Matemático

Registro ordenado y comparativo de resultados estructurales obtenidos mediante distintos programas de simulación computacional

* Resultados del software SAP2000 (Computers & Structures, Inc.)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	M (t*m)	b	Axial max. (t)	Cortante max. (t)	Momento max. (t*m)

* Resultados del Robot Structural (Autodesk)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	M (t*m)	b	Axial max. (t)	Cortante max. (t)	Momento max. (t*m)

* Resultados del software OpenRSE (Universidad de Huánuco)

n	Ux (cm)	Uy (cm)	G (rad)	Rx (t)	Ry (t)	M (t*m)	b	Axial max. (t)	Cortante max. (t)	Momento max. (t*m)

Nota. La ficha de recolección de datos fue elaborada para registrar los resultados de los modelos analizados en la presente investigación.

ANEXO 3

CÓDIGO FUENTE DE LA CLASE PRINCIPAL (ANALIZAR)

```
analizar = class()

function analizar:init(datos)
    self.datos = datos
    self:calcular_longitud_angulo_barra()
    self.k_local = {}
    self:calcular_k_local()
    self.t = {}
    self:calcular_t()
    self.t_traspuesta = {}
    self:calcular_t_traspuesta()
    self.k_global = {}
    self:calcular_k_global()
    self.k_global_ensamblada = {}
    self:calcular_k_global_ensamblada()
    self.gdl_libre = {}
    self.gdl_restringido = {}
    self.gdl_articulado = {}
    self:clasificar_gdl()
    self.KLL = {}
    self.KLR = {}
    self.KRL = {}
    self.KRR = {}
    self:ordenar_k_global_ensamblada()
    self.R1 = {}
    self:ensamblar_R1()
    self.r2_local = {}
    self.r2_global = {}
    self.R2 = {}
    self:ensamblar_R2()
    self.R = {}   self:calcular_R()
    self.RLL = {} self:reducir_R()
    self.KLL_ = {}
    self.DLL = {}
    self:calcular_vector_desplzamiento_gdl_libre()
    self.D_barra = {}
    self.D_local = {}
    self:calcular_vector_desplazamiento_local()
    self.D_global = {}
    self:calcular_vector_desplazamiento_global()
    self.Re_local = {}
    self:calcular_vector_reaccion_local()
    self.Re_global = {}
    self:calcular_vector_reaccion_global()
    self.Qt = {}
    self:calcular_vector_esfuerzos_barra()
```

```
   self.puntos_criticos_cortante = {}
   self:calcular_puntos_criticos_cortante()
   self.puntos_criticos_momento  = {}
   self:calcular_puntos_criticos_momento()
end

function analizar:fuerza_cortante(b,x)
   local barra = self.datos.barra
   local wa = barra.carga[b][1]
   local wb = barra.carga[b][2]
   local L  = barra.longitud[b][1]
   local cv = self.Qt[b][2][1]
   return wa*x+(x^2*(-wa+wb))/(2*L)+cv
end

function analizar:momento_flector(b,x)
   local barra = self.datos.barra
   local wa = barra.carga[b][1]
   local wb = barra.carga[b][2]
   local L  = barra.longitud[b][1]
   local cv = self.Qt[b][2][1]
   local cm = self.Qt[b][3][1]
   return cm-cv*x-(wa*x^2)/2-(x^3*(-wa+wb))/(6*L)
end

function analizar:calcular_puntos_criticos_cortante()
   local barra = self.datos.barra
   for i,k in ipairs(barra.conexion) do
      local wa = barra.carga[i][1]
      local wb = barra.carga[i][2]
      local L  = barra.longitud[i][1]
      local cv = self.Qt[i][2][1]
      local a = (-wa+wb)/(2*L)
      local b = wa
      local c = cv
      local a = string.format("%.10f", a)
      local b = string.format("%.10f", b)
      local c = string.format("%.10f", c)
      math.eval("xmax:=nfMax("..a.."*x^2+"..b.."*x+"..c..",x,0,"..L..")")
      math.eval("xmin:=nfMin("..a.."*x^2+"..b.."*x+"..c..",x,0,"..L..")")
      local xmax = tonumber(math.eval("xmax"))
      local xmin = tonumber(math.eval("xmin"))
      table.insert(self.puntos_criticos_cortante,{xmax,xmin})
      math.eval("DelVar xmax,xmin")
   end

end

function analizar:calcular_puntos_criticos_momento()
   local barra = self.datos.barra
```

```
   for i,k in ipairs(barra.conexion) do
      local wa = barra.carga[i][1]
      local wb = barra.carga[i][2]
      local L  = barra.longitud[i][1]
      local cv = self.Qt[i][2][1]
      local cm = self.Qt[i][3][1]
      local a = -(-wa+wb)/(6*L)
      local b = -wa/2
      local c = -cv
      local d = cm
      local a = string.format("%.10f", a)
      local b = string.format("%.10f", b)
      local c = string.format("%.10f", c)
      local d = string.format("%.10f", d)

math.eval("xmax:=nfMax("..tostring(a).."*x^3+"..tostring(b).."*x^2+"..tostring(c
)).."*x+"..tostring(d)..",x,0,"..tostring(L)..")")

math.eval("xmin:=nfMin("..tostring(a).."*x^3+"..tostring(b).."*x^2+"..tostring(c).
."*x+"..tostring(d)..",x,0,"..tostring(L)..")")
      local xmax = tonumber(math.eval("xmax"))
      local xmin = tonumber(math.eval("xmin"))
      table.insert(self.puntos_criticos_momento,{xmax,xmin})
      math.eval("DelVar xmax,xmin")
   end
end

function analizar:calcular_longitud_angulo_barra()
   local barra = self.datos.barra
   local nodo = self.datos.nodo
   for i,conexion in ipairs(barra.conexion) do
      local ni,nj = conexion[1],conexion[2]
      local xi,yi = nodo.coordenada[ni][1],nodo.coordenada[ni][2]
      local xj,yj = nodo.coordenada[nj][1],nodo.coordenada[nj][2]
      local longitud = math.sqrt((xj-xi)^2+(yj-yi)^2)
      table.insert(self.datos.barra.longitud,{longitud})
      local angulo_radianes = math.atan2(yj-yi, xj-xi)
      local angulo_grados = math.deg(angulo_radianes)
      if angulo_grados < 0 then
         angulo_grados = angulo_grados + 360
      end
      table.insert(self.datos.barra.angulo,{angulo_grados})
   end
end

function analizar:matriz_rigidez_na_na(E,A,I,L)
   local K = {
      {E*A/L,0,0,-E*A/L,0,0},
      {0,12*E*I/L^3,6*E*I/L^2,0,-12*E*I/L^3,6*E*I/L^2},
      {0,6*E*I/L^2,4*E*I/L,0,-6*E*I/L^2,2*E*I/L},
```

```
      {-E*A/L,0,0,E*A/L,0,0},
      {0,-12*E*I/L^3,-6*E*I/L^2,0,12*E*I/L^3,-6*E*I/L^2},
      {0,6*E*I/L^2,2*E*I/L,0,-6*E*I/L^2,4*E*I/L}
   }
   return K
end

function analizar:matriz_rigidez_a_na(E,A,I,L)
   local K = {
    {E*A/L,0,0,-E*A/L,0,0},
    {0,3*E*I/L^3,0,0,-3*E*I/L^3,3*E*I/L^2},
    {0,0,0,0,0,0},
    {-E*A/L,0,0,E*A/L,0,0},
    {0,-3*E*I/L^3,0,0,3*E*I/L^3,-3*E*I/L^2},
    {0,3*E*I/L^2,0,0,-3*E*I/L^2,3*E*I/L}
   }
   return K
end

function analizar:matriz_rigidez_na_a(E,A,I,L)
   local K = {
      {E*A/L,0,0,-E*A/L,0,0},
      {0,3*E*I/L^3,3*E*I/L^2,0,-3*E*I/L^3,0},
      {0,3*E*I/L^2,3*E*I/L,0,-3*E*I/L^2,0},
      {-E*A/L,0,0,E*A/L,0,0},
      {0,-3*E*I/L^3,-3*E*I/L^2,0,3*E*I/L^3,0},
      {0,0,0,0,0,0}
   }
   return K
end

function analizar:matriz_rigidez_a_a(E,A,L)
   local K = {
      {E*A/L,0,0,-E*A/L,0,0},
      {0,0,0,0,0,0},
      {0,0,0,0,0,0},
      {-E*A/L,0,0,E*A/L,0,0},
      {0,0,0,0,0,0},
      {0,0,0,0,0,0}
   }
   return K
end

function analizar:calcular_k_local()
   local barra = self.datos.barra
   for i, k in ipairs(barra.conexion) do
      local A,I = barra.propiedad[i][1],barra.propiedad[i][2]
      local E,L = barra.propiedad[i][3],barra.longitud[i][1]
      local ci,cj = barra.contorno[i][1],barra.contorno[i][2]
      local k_local
```

```
      if not ci and not cj then
        k_local = self:matriz_rigidez_na_na(E,A,I,L)
      elseif ci and not cj then
        k_local = self:matriz_rigidez_a_na(E,A,I,L)
      elseif not ci and cj then
        k_local = self:matriz_rigidez_na_a(E,A,I,L)
      elseif ci and cj  then
        k_local = self:matriz_rigidez_a_a(E,A,L)
      end
      table.insert(self.k_local, k_local)
   end
end

function analizar:calcular_t()
   local barra = self.datos.barra
   local nodo  = self.datos.nodo
   for i, conexion in ipairs(barra.conexion) do
      local theta  = math.rad(barra.angulo[i][1])
      local beta_i = math.rad(nodo.restriccion[conexion[1]][2])
      local beta_j = math.rad(nodo.restriccion[conexion[2]][2])
      local t = {
         {math.cos(theta-beta_i),-math.sin(theta-beta_i),0,0,0,0},
         {math.sin(theta-beta_i), math.cos(theta-beta_i),0,0,0,0},
         {0,0,1,0,0,0},
         {0,0,0,math.cos(theta-beta_j),-math.sin(theta-beta_j),0},
         {0,0,0,math.sin(theta-beta_j),math.cos(theta-beta_j),0},
         {0,0,0,0,0,1}
      }
      table.insert(self.t,t)
   end
end


function analizar:trasponer_matriz(a)
   local a_traspuesta = {}
   for i = 1, #a[1] do
      a_traspuesta[i] = {}
      for j = 1, #a do
         a_traspuesta[i][j] = a[j][i]
      end
   end
   return a_traspuesta
end

function analizar:calcular_t_traspuesta()
   local barra = self.datos.barra
   for i, k in ipairs(barra.conexion) do
      local t_traspuesta = self:trasponer_matriz(self.t[i])
      table.insert(self.t_traspuesta,t_traspuesta)
   end
```

```
end


function analizar:multiplicar_matriz(a,b)
   local producto = {}
   for i = 1, #a do
      producto[i] = {}
      for j = 1, #b[1] do
         producto[i][j] = 0
         for k = 1, #a[1] do
           producto[i][j] = producto[i][j] + a[i][k] * b[k][j]
         end
      end
   end
   return producto
end

function analizar:calcular_k_global()
   local barra = self.datos.barra
   for i, k in ipairs(barra.conexion) do
      local k_local = self.k_local[i]
      local t = self.t[i]
      local t_traspuesta = self.t_traspuesta[i]
      local temporal = self:multiplicar_matriz(t,k_local)
      local k_global = self:multiplicar_matriz(temporal,t_traspuesta)
      table.insert(self.k_global,k_global)
   end
end


function analizar:calcular_k_global_ensamblada()
   local nodo  = self.datos.nodo
   local barra = self.datos.barra
   for i = 1, #nodo.coordenada * 3 do
      self.k_global_ensamblada[i] = {}
      for j = 1, #nodo.coordenada * 3 do
         self.k_global_ensamblada[i][j] = 0
      end
   end
   for i, conexion in ipairs(barra.conexion) do
      local k_global = self.k_global[i]
      local ni = conexion[1]
      local nj = conexion[2]
      for i = 1, 3 do
         for j = 1, 3 do
            local fila,columna
            fila = (ni - 1) * 3 + i
            columna = (ni - 1) * 3 + j
            self.k_global_ensamblada[fila][columna]                    =
self.k_global_ensamblada[fila][columna] + k_global[i][j]
```

```
            fila = (ni - 1) * 3 + i
            columna = (nj - 1) * 3 + j
            self.k_global_ensamblada[fila][columna]                                    =
self.k_global_ensamblada[fila][columna] + k_global[i][j + 3]
            fila = (nj - 1) * 3 + i
            columna = (ni - 1) * 3 + j
            self.k_global_ensamblada[fila][columna]                                    =
self.k_global_ensamblada[fila][columna] + k_global[i + 3][j]
            fila = (nj - 1) * 3 + i
            columna = (nj - 1) * 3 + j
            self.k_global_ensamblada[fila][columna]                                    =
self.k_global_ensamblada[fila][columna] + k_global[i + 3][j + 3]
         end
      end
   end
end

function analizar:clasificar_gdl()
   restricciones = {
      {true,  true,  true},
      {false, false, false},
      {false, false, true},
      {true,  false, true},
      {false, true,  true},
      {true,  true,  false},
      {true,  false, false},
      {false, true,  false}
   }
   local nodo  = self.datos.nodo
   local barra = self.datos.barra
   for i, k in ipairs(nodo.coordenada) do
      local cumpleCondicion = true
      for j, conexion in ipairs(barra.conexion) do
         local ni = conexion[1]
         local nj = conexion[2]
         local ci = barra.contorno[j][1]
         local cj = barra.contorno[j][2]
         local articulado
         if ni == i then
            articulado = ci
         elseif nj == i then
            articulado = cj
         else articulado = nil
         end
         if not articulado and articulado ~= nil then
            cumpleCondicion = false
         end
      end
      if cumpleCondicion then
         local gdl = ((i - 1) * 3) + 3
```

```
            table.insert(self.gdl_articulado, gdl)
        end
    end
    for i, restriccion in ipairs(nodo.restriccion) do
        local condicion = restriccion[1]
        local baseGDL = (i - 1) * 3
        local restriccion = restricciones[condicion]
        for gdl = 1, 3 do
            local gdlActual = baseGDL + gdl
            local esArticulado = false
            for _, gdlArticulado in ipairs(self.gdl_articulado) do
                if gdlArticulado == gdlActual then
                    esArticulado = true
                    break
                end
            end
            if esArticulado or not restriccion[gdl] then
                table.insert(self.gdl_restringido, gdlActual)
            else
                table.insert(self.gdl_libre, gdlActual)
            end
        end
    end
end

function analizar:ordenar_k_global_ensamblada()
    local nuevoOrdenGDL = {}
    for _, gdl in ipairs(self.gdl_libre) do table.insert(nuevoOrdenGDL, gdl) end
    for _, gdl in ipairs(self.gdl_restringido) do table.insert(nuevoOrdenGDL,
gdl) end
    local nLibres = #self.gdl_libre
    for i, gdl_i in ipairs(nuevoOrdenGDL) do
        for j, gdl_j in ipairs(nuevoOrdenGDL) do
            local valor = self.k_global_ensamblada[gdl_i][gdl_j]
            if i <= nLibres then
                if j <= nLibres then
                    self.KLL[i] = self.KLL[i] or {}
                    self.KLL[i][j] = valor
                else
                    self.KLR[i] = self.KLR[i] or {}
                    self.KLR[i][j - nLibres] = valor
                end
            else
                if j <= nLibres then
                    self.KRL[i - nLibres] = self.KRL[i - nLibres] or {}
                    self.KRL[i - nLibres][j] = valor
                else
                    self.KRR[i - nLibres] = self.KRR[i - nLibres] or {}
                    self.KRR[i - nLibres][j - nLibres] = valor
                end
```

```
            end
        end
    end
end

function analizar:ensamblar_R1()
    local nodo = self.datos.nodo
    for i = 1, #nodo.carga do
        local contador = (i - 1) * 3
        self.R1[contador + 1] = {nodo.carga[i][1]}
        self.R1[contador + 2] = {nodo.carga[i][2]}
        self.R1[contador + 3] = {nodo.carga[i][3]}
    end
end

function analizar:empotramiento_perfecto_na_na(L,Wi,Wj)
    if Wi == 0 and Wj == 0 then return 0,0,0,0 end
    local Mi = -L^2/60 * (3*Wi + 2*Wj)
    local Mj = L^2/60 * (2*Wi + 3*Wj)
    local Vi = -L/20 * (7*Wi + 3*Wj)
    local Vj = -L/20 * (3*Wi + 7*Wj)
    return Vi,Mi,Vj,Mj
end

function analizar:empotramiento_perfecto_a_na(L,Wi,Wj)
    if Wi == 0 and Wj == 0 then return 0,0,0,0 end
    local Vi = -(L/120)*(33*Wi+12*Wj)
    local Vj = -(L/120)*(27*Wi+48*Wj)
    local Mi = 0
    local Mj = (L^2/120)*(7*Wi+8*Wj)
    return Vi,Mi,Vj,Mj
end

function analizar:empotramiento_perfecto_na_a(L,Wi,Wj)
    if Wi == 0 and Wj == 0 then return 0,0,0,0 end
    local entrada_wi = Wi
    local entrada_wj = Wj
    local Wi = entrada_wj
    local Wj = entrada_wi
    local Vi = -(L/120)*(27*Wi+48*Wj)
    local Vj = -(L/120)*(33*Wi+12*Wj)
    local Mi = -(L^2/120)*(7*Wi+8*Wj)
    local Mj = 0
    return Vi,Mi,Vj,Mj
end

function analizar:empotramiento_perfecto_a_a(L,Wi,Wj)
  if Wi == 0 and Wj == 0 then return 0,0,0,0 end
  local Vi = -(L*(2*Wi+Wj))/6
  local Vj = -(L*(Wi+2*Wj))/6
```

```
  local Mi = 0
  local Mj = 0
  return Vi,Mi,Vj,Mj
end

function analizar:ensamblar_R2()
   local barra = self.datos.barra
   local nodo  = self.datos.nodo
   for i,carga in ipairs(barra.carga) do
      local l  = barra.longitud[i][1]
      local wi,wj = carga[1],carga[2]
      local ci,cj = barra.contorno[i][1],barra.contorno[i][2]
      local vi, mi, vj, mj
      if ci and cj then
         vi, mi, vj, mj = self:empotramiento_perfecto_a_a(l,wi,wj)
      elseif ci  and not cj then
         vi, mi, vj, mj = self:empotramiento_perfecto_a_na(l,wi,wj)
      elseif not ci and cj then
         vi, mi, vj, mj = self:empotramiento_perfecto_na_a(l,wi,wj)
      else
         vi, mi, vj, mj = self:empotramiento_perfecto_na_na(l,wi,wj)
      end
      local r2_local={{0},{vi},{mi},{0},{vj},{mj}}
      table.insert(self.r2_local,r2_local)
   end
   for i,k in ipairs(barra.conexion) do
      local t = self.t[i]
      local r2_local = self.r2_local[i]
      local r2_global = self:multiplicar_matriz(t,r2_local)
      table.insert(self.r2_global,r2_global)
   end
   for i = 1, #nodo.coordenada * 3 do
      self.R2[i] = {0}
   end
   for i, conexion in ipairs(barra.conexion) do
      local ni, nj = conexion[1], conexion[2]
      local cargasBarra = self.r2_global[i]
      for j = 1, 6 do
         local gdl_global
         if j <= 3 then
            gdl_global = (ni - 1) * 3 + j
         else
            gdl_global = (nj - 1) * 3 + (j - 3)
         end
         self.R2[gdl_global][1] = self.R2[gdl_global][1] + cargasBarra[j][1]
      end
   end
end

function analizar:restar_matriz(a,b)
```

```
    local resta = {}
    for i = 1, #a do
        resta[i] = {}
        for j = 1, #a[1] do
            resta[i][j] = a[i][j] - b[i][j]
        end
    end
    return resta
end

function analizar:calcular_R()
    self.R = self:restar_matriz(self.R1,self.R2)
end

function analizar:reducir_R()
    for i, gdl in ipairs(self.gdl_libre) do
        table.insert(self.RLL, self.R[gdl])
    end
end

function analizar:invertir_matriz(A)
    local n = #A
    local copiaA = {}
    for i = 1, n do
        copiaA[i] = {}
        for j = 1, n do
            copiaA[i][j] = A[i][j]
        end
    end
    local I = {}
    for i = 1, n do
        I[i] = {}
        for j = 1, n do
            I[i][j] = (i == j) and 1 or 0
        end
    end
    for i = 1, n do
        if copiaA[i][i] == 0 then
            error("La matriz contiene un cero en la diagonal.")
        end
        for j = 1, n do
            if i ~= j then
                local ratio = copiaA[j][i] / copiaA[i][i]
                for k = 1, n do
                    copiaA[j][k] = copiaA[j][k] - ratio * copiaA[i][k]
                    I[j][k] = I[j][k] - ratio * I[i][k]
                end
            end
        end
    end
```

```
    for i = 1, n do
        local divisor = copiaA[i][i]
        for j = 1, n do
            I[i][j] = I[i][j] / divisor
        end
    end
    return I
end

function analizar:calcular_vector_desplzamiento_gdl_libre()
    local KLL_inversa = self:invertir_matriz(self.KLL)
    self.KLL_ = KLL_inversa
    self.DLL = self:multiplicar_matriz(KLL_inversa,self.RLL)
end

function analizar:calcular_vector_desplazamiento_local()
    for i = 1, #self.gdl_libre + #self.gdl_restringido do
        table.insert(self.D_local, {0})
    end
    for i, gdlLibre in ipairs(self.gdl_libre) do
        self.D_local[gdlLibre] = self.DLL[i]
    end
end

function analizar:calcular_vector_desplazamiento_global()
   local nodo = self.datos.nodo
   for i = 1, #nodo.coordenada do
      local baseIndex = (i - 1) * 3
      local u1 = self.D_local[baseIndex + 1][1]
      local u2 = self.D_local[baseIndex + 2][1]
      local u3 = self.D_local[baseIndex + 3][1]
      local temp1 = {{u1},{u2},{u3}}
      local angulo_grados = nodo.restriccion[i][2]
      local angulo_radianes = math.rad(angulo_grados)
      local m = {
         {math.cos(angulo_radianes),-math.sin(angulo_radianes),0},
         {math.sin(angulo_radianes),math.cos(angulo_radianes),0},
         {0,0,1},
      }
      local temp2 = self:multiplicar_matriz(m,temp1)
      self.D_global[baseIndex + 1] = {temp2[1][1]}
      self.D_global[baseIndex + 2] = {temp2[2][1]}
      self.D_global[baseIndex + 3] = {temp2[3][1]}
   end
end

function analizar:calcular_vector_reaccion_local()
    local temporal =
self:multiplicar_matriz(self.k_global_ensamblada,self.D_local)
    self.Re_local = self:restar_matriz(temporal,self.R)
```

```
end

function analizar:calcular_vector_reaccion_global()
   local nodo = self.datos.nodo
   for i = 1, #nodo.coordenada do
      local baseIndex = (i - 1) * 3
      local u1 = self.Re_local[baseIndex + 1][1]
      local u2 = self.Re_local[baseIndex + 2][1]
      local u3 = self.Re_local[baseIndex + 3][1]
      local temp1 = {{u1},{u2},{u3}}
      local angulo_grados = nodo.restriccion[i][2]
      local angulo_radianes = math.rad(angulo_grados)
      local m = {
         {math.cos(angulo_radianes),-math.sin(angulo_radianes),0},
         {math.sin(angulo_radianes),math.cos(angulo_radianes),0},
         {0,0,1},
      }
      local temp2 = self:multiplicar_matriz(m,temp1)
      self.Re_global[baseIndex + 1] = {temp2[1][1]}
      self.Re_global[baseIndex + 2] = {temp2[2][1]}
      self.Re_global[baseIndex + 3] = {temp2[3][1]}
   end
end

function analizar:sumar_matriz(a,b)
    local resta = {}
   for i = 1, #a do
      resta[i] = {}
      for j = 1, #a[1] do
         resta[i][j] = a[i][j] + b[i][j]
      end
   end
   return resta
end

function analizar:calcular_vector_esfuerzos_barra()
   local barra = self.datos.barra
   for i, barra in ipairs(barra.conexion) do
      local gdl_i_inicio = (barra[1] - 1) * 3 + 1
      local gdl_j_inicio = (barra[2] - 1) * 3 + 1
      local D_ = {}
      for gdl = gdl_i_inicio, gdl_i_inicio + 2 do
         table.insert(D_, {self.D_local[gdl][1]})
      end
      for gdl = gdl_j_inicio, gdl_j_inicio + 2 do
         table.insert(D_, {self.D_local[gdl][1]})
      end
      self.D_barra[i] = D_
      local D_transformado = self:multiplicar_matriz(self.t_traspuesta[i], D_)
      local Qt_actual = self:multiplicar_matriz(self.k_local[i], D_transformado)
```

```
        local Qt_sumado = self:sumar_matriz(Qt_actual, self.r2_local[i])
        table.insert(self.Qt, Qt_sumado)
    end
end
```