ANÁLISIS MATRICIAL DE ESTRUCTURAS 2D MEDIANTE EL MÉTODO DE RIGIDEZ EN PYTHON

Orias E.[a]

[a] Docente curso Sap 2000 de la Facultad de Ingeniería Civil (USFX), Destacamento 317, Ex Campus REFISUR,
573, Sucre, Bolivia. E-mail: eoriasf@gmail.com.
[b] Master en cálculo de estructuras de obras civiles, EADIC, 2022

Recibido: 1/10/2025 Aceptado: 15/10/2025 Publicado: 10/11/2025

RESUMEN

El análisis estructural es una disciplina fundamental en la ingeniería civil, orientada a garantizar la seguridad, estabilidad y funcionalidad de las estructuras frente a cargas gravitatorias, térmicas y dinámicas. Entre los métodos numéricos más utilizados destaca el método de rigidez, que permite modelar y resolver sistemas estructurales complejos mediante operaciones matriciales.

Este artículo presenta el desarrollo y la implementación del método de rigidez en Python, utilizando librerías como NumPy para cálculos matriciales y Matplotlib para la visualización gráfica de desplazamientos, reacciones y fuerzas internas. Se detallan los fundamentos teóricos del método, la estructura computacional del algoritmo y un ejemplo práctico de análisis de una armadura plana 2D.

Para facilitar la comprensión del código, la metodología se organiza en tres fases claramente diferenciadas: Preproceso (lectura de datos y definición del modelo), Proceso (ensamblaje de matrices, resolución de desplazamientos y cálculo de fuerzas internas) y Postproceso (generación de resultados, reportes y gráficos).

Los resultados obtenidos evidencian la eficiencia, precisión y versatilidad de Python como herramienta para la simulación y el análisis de sistemas estructurales complejos, facilitando tanto el cálculo numérico como la interpretación gráfica de los resultados.

Palabras clave: Análisis estructural, Método de rigidez, Armaduras 2D, Python

ABSTRACT

Structural analysis is a fundamental discipline in civil engineering, aimed at ensuring the safety, stability, and functionality of structures under gravitational, thermal, and dynamic loads. Among the most widely used numerical methods, the stiffness method stands out for its ability to model and solve complex structural systems using matrix operations.

This paper presents the development and implementation of the stiffness method in Python, using libraries such as NumPy for matrix calculations and Matplotlib for the graphical visualization of displacements, reactions, and internal forces. The theoretical foundations of the method, the computational structure of the algorithm, and a practical example of a 2D planar truss analysis are detailed.

To facilitate understanding of the code, the methodology is organized into three clearly differentiated phases: Preprocessing (data reading and model definition), Processing (assembly of matrices, solution of displacements, and calculation of internal forces), and Postprocessing (generation of results, reports, and graphs).

The results demonstrate the efficiency, accuracy, and versatility of Python as a tool for the simulation and analysis of complex structural systems, facilitating both numerical computation and graphical interpretation of results.

Key words: Structural analysis, Stiffness method, 2D trusses, Python

INTRODUCCIÓN

El método de rigidez es fundamental en el análisis matricial de estructuras y permite formular de manera sistemática las ecuaciones de equilibrio mediante el ensamblaje de matrices de rigidez de los elementos, obteniendo la matriz global que relaciona desplazamientos y fuerzas nodales. Este enfoque es especialmente útil para estructuras planas 2D, vigas y armaduras, así como para sistemas estructurales complejos.

En la actualidad, la creciente complejidad de las estructuras requiere herramientas computacionales eficientes. Python, con librerías como NumPy y Matplotlib, ofrece un entorno flexible para implementar algoritmos de análisis estructural, realizar cálculos matriciales y generar visualizaciones gráficas de desplazamientos, reacciones y esfuerzos internos.

El objetivo de este trabajo es presentar la formulación teórica y la implementación del método de rigidez en Python, mostrando un ejemplo práctico de una armadura plana 2D. Para facilitar la comprensión, el procedimiento se organiza en tres fases claramente diferenciadas: Preproceso (definición del modelo y lectura de datos), Proceso (ensamblaje de matrices, resolución de desplazamientos y cálculo de fuerzas internas) y Postproceso (generación de resultados, reportes y gráficos). Este enfoque permite vincular la teoría estructural con la programación científica de manera práctica y didáctica.

ANALISIS MATRICIAL DE ESTRUCTURAS

(METODO DE RIGIDEZ)

1. Idealización de la estructura

- Definir la geometría: nodos $i = 1, \dots, N_n$ y elementos $e = 1, \dots, N_e$
- Tipo de elemento armadura 2D
- Cada elemento se caracteriza por sus propiedades físicas:

  E_e = Módulo de elasticidad
  A_e = Área de la sección, Le (longitud)
  L_e = Longitud del elemento

$$L_e = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

- Definir grados de libertad (GDL) por nodo.

$$\{D_i\} = \{^{u_i}_{v_i}\}$$

Donde: u_i y v_i representan los desplazamientos en la dirección x e y respectivamente

- Para una armadura plana con N_n nodos:

$$N_{GDL} = 2N_n$$

2. Matriz de rigidez de cada elemental en coordenadas locales

En el sistema local (x', y'), que solo puede deformarse axialmente, por tanto la relación fuerza-desplazamiento será:

$$\{f^{(e)}\}_{local} = [k^e]_{local}\{d^{(e)}\}_{local}$$

Para cada tipo de elemento se obtiene una matriz de rigidez local $[k^e]_{local}$

$$[k^e]_{local} = \frac{A_e * E_e}{L_e}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Los desplazamientos locales están dados por:

$$\{d^{(e)}\}_{local} = \begin{Bmatrix} u_i' \\ u_i' \end{Bmatrix}$$

3. Transformación a coordenadas globales

Como la orientación de cada barra es arbitraria, es necesario la transformar los desplazamientos y fuerzas locales (x', y') al sistema global (x, y),

Los cosenos directores del eje de la barra son:

$$\lambda_x = cos\theta_x = \frac{x_f - x_i}{L_e}$$

$$\lambda_y = sen\theta_y = \frac{y_f - y_i}{L_e}$$

La matriz de transformación que relaciona desplazamientos locales y globales será $[T^e]$

$$\{d^{(e)}\}_{local} = [T^{(e)}]\{D^{(e)}\}_{global}$$

Donde:

$$[T^{(e)}] = \begin{bmatrix} \lambda_x & \lambda_y & 0 & 0 \\ 0 & 0 & \lambda_x & \lambda_y \end{bmatrix}$$

Donde:

Transformación de rigidez

$$[k^{(e)}]_{global} = [T^{(e)}]^T [k^{(e)}]_{local}[T^{(e)}]$$

Sustituyendo esta relación en la ecuación local:

$$\{f^{(e)}\}_{global} = [T^{(e)}]^T [k^{(e)}]_{local}[T^{(e)}]\{D^{(e)}\}_{global}$$

La matriz de rigidez elemental en coordenadas globales será:

$$[k^{(e)}]_{global} = \frac{A_e E_e}{L_e}\begin{bmatrix} \lambda_x{}^2 & \lambda_x\lambda_y & -\lambda_x{}^2 & -\lambda_x\lambda_y \\ \lambda_x\lambda_y & \lambda_y{}^2 & -\lambda_x\lambda_y & -\lambda_y{}^2 \\ -\lambda_x{}^2 & -\lambda_x\lambda_y & \lambda_x{}^2 & \lambda_x\lambda_y \\ -\lambda_x\lambda_y & -\lambda_y{}^2 & \lambda_x\lambda_y & \lambda_y{}^2 \end{bmatrix}$$

4. Ensamblaje de la matriz de rigidez y vectores de carga

El conjunto estructural se modela mediante la ecuación matricial general:

$$[K]\{D\} = \{F\}$$

Donde:

$[K]$ es la matriz de rigidez global de dimensiones $(2N_n * 2N_n)$

$\{D\}$ es el vector global de desplazamientos.

$\{F\}$ es el vector global de fuerzas externas

El ensamblaje se realiza superponiendo las condiciones de cada elemento mediante la matriz de conectividad $A^{(e)}$:

$$[K] = \sum_{e=1}^{N_e} A^{(e)}[k^{(e)}]_{global}A^{(e)^T}$$

Donde:

$[K]$=Matriz de rigidez global

$A^{(e)}$= Matriz de conexión o de extensión, es la matriz binaria que mapea los grados de libertad locales y globales.

De forma análoga, se ensamblar también el vector global de cargas

$$\{\boldsymbol{F}\} = \sum_{\boldsymbol{e=1}}^{\boldsymbol{N_e}} A^{(e)}\ \{f^{(e)}\}$$

5. Condiciones de contorno (apoyos) y fraccionar el sistema

Para imponer los apoyos (restricciones), se fraccionan la ecuación global en función de los grados de libertad:

$$\begin{Bmatrix}\{F_a\}\\{F_b}\end{Bmatrix} = \begin{bmatrix}[K_{aa}] & [K_{ab}]\\[K_{ba}] & [K_{bb}]\end{bmatrix}\begin{Bmatrix}\{D_a\}\\{D_b}\end{Bmatrix}$$

Donde:

$\{F_a\}$=Subvector de fuerzas de los GDL libres (conocidos)

$\{D_b\}$=Subvector de desplazamientos desconocidos, de los GDL libres

$\{F_b\}$=Subvector de fuerzas desconocidas corresponde a las reacciones en los apoyos

$\{D_a\}$=Subvector de desplazamientos conocidos son todos nulos ($D_a = 0$), se obtiene:

$$[K_{bb}]\{D_b\} = \{F_b\}$$

6. Resolución del sistema de ecuaciones y cálculo de desplazamientos

Como los elementos de $\{F_a\}$ son conocidos se calcularan los desplazamientos de los grados de libertad no restringidos $\{D_b\}$

$$\{D_b\} = [K_{bb}]^{-1}\{F_b\}$$

7. Calculo de reacciones

Las **reacciones** en los apoyos se calculan como:

$$\{F_a\} = [K_{ab}]\{D_b\}$$

8. Calculo de fuerzas internas

$$[f^{(e)}]_{local} = [k^{(e)}]_{local}[T^{(e)}][D^{(e)}]_{global}$$

9. Naturaliza lineal e hiperestática del análisis estructural

El sistema estructural resultante es un sistema lineal de ecuaciones algebraica, de orden $n = 2N_n$

Si el número de ecuaciones independientes coincide con el número de incógnitas, la

estructura es isostática; de lo contrario, es hiperestática y el método de rigidez la resuelve de forma directa sin necesidad de ecuaciones de compatibilidad adicionales, ya que el equilibro y la compatibilidad están implícitos en la formulación matricial.

El carácter lineal del problema implica que:

$[K]$ es simétrica y definida positiva y los desplazamientos son proporcionales a las cargas aplicadas.

DESARROLLO COMPUTACIONAL DEL CODIGO EN PYTHON

El desarrollo computacional del código en Python implementa de manera sistemática el método de rigidez para el análisis matricial de armaduras planas 2D.

El programa se estructura en tres etapas principales:

1. Preproceso

En esta etapa se realiza la definición del modelo estructural y la preparación de los datos de entrada.

Se leen los archivos que contienen la información geométrica (coordenadas nodales), la conectividad de los elementos (nodos iniciales y finales), las propiedades mecánicas (área y módulo de elasticidad), las condiciones de apoyo y las cargas aplicadas.

Con estos datos, el programa calcula automáticamente las longitudes y ángulos de inclinación de cada barra, necesarios para la construcción de las matrices de rigidez locales y su transformación al sistema global.

El preproceso, en esencia, convierte la descripción física de la estructura en una representación matemática y computacional, estableciendo los grados de libertad y la topología del modelo.

2. Proceso

En esta fase se desarrolla el análisis matricial propiamente dicho, siguiendo el método de rigidez.

Para cada barra, se forma la matriz de rigidez local, que expresa la relación entre las fuerzas y desplazamientos en coordenadas del elemento.

Mediante una matriz de transformación, estas matrices se convierten a coordenadas globales y se ensamblan para construir la matriz de rigidez global del sistema.

A continuación, se aplican las condiciones de contorno que representan los apoyos o restricciones, reduciendo el sistema a los grados de libertad libres.

El sistema lineal resultante se resuelve para obtener los desplazamientos nodales, lo que permite conocer el estado deformado de la estructura.

Con estos desplazamientos, se calculan las reacciones en los apoyos y las fuerzas internas de cada elemento, completando el equilibrio estructural.

3. Postproceso

El postproceso se dedica a la evaluación e interpretación de los resultados obtenidos del análisis.

Se generan reportes numéricos con desplazamientos, reacciones y esfuerzos axiales en las barras.

Asimismo, el programa produce representaciones gráficas de la estructura original y deformada, mostrando también las cargas y apoyos aplicados.

Para mantener la coherencia visual, se implementa un escalado automático que ajusta el tamaño de textos, símbolos y

deformaciones según la dimensión de la estructura.

Esta fase permite validar los resultados del modelo y facilita su interpretación visual y técnica.

RESULTADOS OBTENIDO CON EL PROGRAMA ELABORADO EN PYTHON

PREPROCESO: Lectura de datos y definición del modelo

Determinar las fuerzas que actual en todos los elementos de la armadura que se muestra en la figura 1 (ejemplo extraído del libro de estática de R.C. Hibbeler para verificar los resultados)

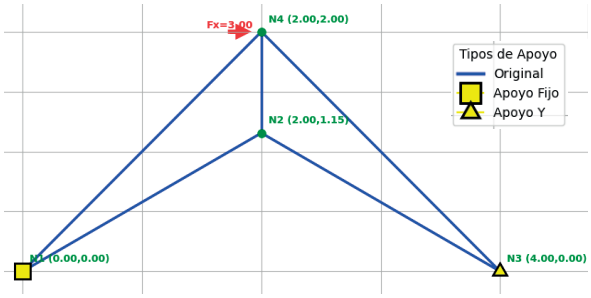

Figura 1: Armadura a analizar (numeración de nodos y coordenadas)

Contenido de datos de archivo con el nombre: datos2D_ejemplo_1.txt

```
TIPO:
2D
#Nodos: ID  X Y Z
NODOS:
1,0,0
2,2,1.1547
3,4,0
4,2,2
#Barras: ID Area  Mod_Elasticidad Nodo_Inicial
Nodo_Final
BARRAS:
1,1,1,1,2
2,1,1,2,3
3,1,1,3,4
4,1,1,1,4
5,1,1,2,4

#Apoyo: Nodo DX DY (1 = Restringido, 0 = Libre)
APOYOS:
1,1,1
3,0,1

#Carga: Nodo FX FY
CARGAS:
4,3,0

# Opciones de grafico (1 = Mostrar, 0 = Ocultar)
GRAFICAS:
nodos=1
coordenadas_nudos=1
barras=1
nombre_barras=1
cargas=1
apoyos=1
grados_de_libertad=1
deformada=0
tension_compresion=0
reacciones=0
imprimir_resultados=1
```

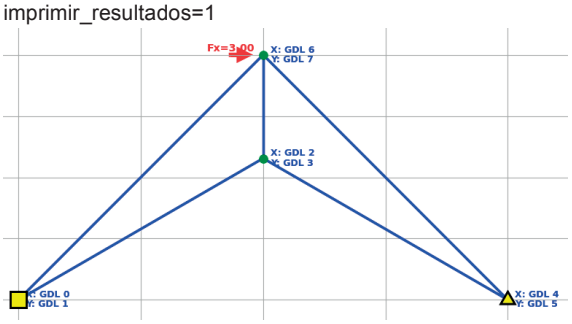

Figura 2: grados de libertad definidos por el programa

Conexiones de barras (Nodo Inicial -> Nodo Final):

Barra 1: Nodo 1 -> Nodo 2
Barra 2: Nodo 2 -> Nodo 3
Barra 3: Nodo 3 -> Nodo 4
Barra 4: Nodo 1 -> Nodo 4
Barra 5: Nodo 2 -> Nodo 4

PROCESO: Ensamblaje de matrices, resolución de desplazamientos, reacciones y fuerzas internas

Reporte del programa: resultado2D_detallado.txt

Longitudes y ángulos de barras:

Barra 1: Longitud L = 2.3094, Ángulo β = 30.00°
Barra 2: Longitud L = 2.3094, Ángulo β = -30.00°
Barra 3: Longitud L = 2.8284, Ángulo β = 135.00°
Barra 4: Longitud L = 2.8284, Ángulo β = 45.00°
Barra 5: Longitud L = 0.8453, Ángulo β = 90.00°

Matrices de rigidez global transformadas de cada barra:

Barra 1 (Nodo 1 -> Nodo 2):

```
 0.324760   0.187500  -0.324760  -0.187500
 0.187500   0.108253  -0.187500  -0.108253
-0.324760  -0.187500   0.324760   0.187500
-0.187500  -0.108253   0.187500   0.108253

Barra  2 (Nodo 2 -> Nodo 3):
 0.324760  -0.187500  -0.324760   0.187500
-0.187500   0.108253   0.187500  -0.108253
-0.324760   0.187500   0.324760  -0.187500
 0.187500  -0.108253  -0.187500   0.108253

Barra  3 (Nodo 3 -> Nodo 4):
 0.176777  -0.176777  -0.176777   0.176777
-0.176777   0.176777   0.176777  -0.176777
-0.176777   0.176777   0.176777  -0.176777
 0.176777  -0.176777  -0.176777   0.176777

Barra  4 (Nodo 1 -> Nodo 4):
 0.176777   0.176777  -0.176777  -0.176777
 0.176777   0.176777  -0.176777  -0.176777
-0.176777  -0.176777   0.176777   0.176777
-0.176777  -0.176777   0.176777   0.176777
Barra  5 (Nodo 2 -> Nodo 4):
 0.000000   0.000000  -0.000000  -0.000000
 0.000000   1.183012  -0.000000  -1.183012
-0.000000  -0.000000   0.000000   0.000000
-0.000000  -1.183012   0.000000   1.183012

Matriz global ensamblada K_global:
 0.501  0.364 -0.324 -0.187  0.000  0.00 -0.176 -0.176
 0.364  0.285 -0.187 -0.108  0.000  0.00 -0.176 -0.176
-0.324 -0.187  0.649  0.000 -0.324  0.187 -0.000 -0.00
-0.187 -0.108  0.000  1.399  0.187 -0.108 -0.000 -1.183
 0.000  0.00  -0.324  0.187  0.501 -0.364 -0.176  0.176
 0.000  0.000  0.187 -0.108 -0.364  0.285  0.176 -0.176
-0.176 -0.176  0.000  0.000 -0.176  0.176  0.353  0.000
-0.176 -0.176  0.000 -1.183  0.176 -0.176  0.000  1.536

Desplazamientos nodales D:
  Desplazamiento Nodo   1: Dx = 0.0000, Dy = 0.000
  Desplazamiento Nodo   2: Dx = 46.422, Dy = -61.47
  Desplazamiento Nodo   3: Dx = 92.844, Dy = 0.000
  Desplazamiento Nodo   4: Dx = 54.907, Dy = -58.01

Reacciones en apoyos:
  Grado de libertad   1: Reacción =    -3.000000
  Grado de libertad   2: Reacción =    -1.500000
  Grado de libertad   6: Reacción =     1.500000

Fuerzas internas en barras (N):
  Elemento   1 (Nodo 1 -> Nodo 2): N =     4.098073
  Elemento   2 (Nodo 2 -> Nodo 3): N =     4.098073
  Elemento   3 (Nodo 3 -> Nodo 4): N =    -5.019095
  Elemento   4 (Nodo 1 -> Nodo 4): N =    -0.776454
  Elemento   5 (Nodo 2 -> Nodo 4): N =     4.098072
```

POSTPROCESO: Resultados, deformada e la estructura, desplazamientos, reacciones y fuerzas internas

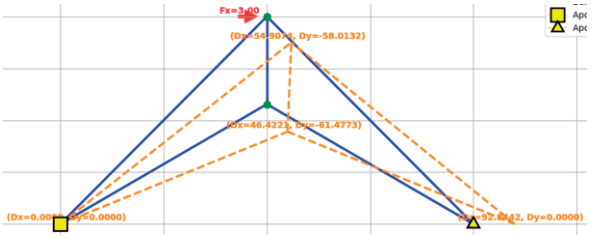

Figura 3: Deformada de la estructura (desplazamientos)

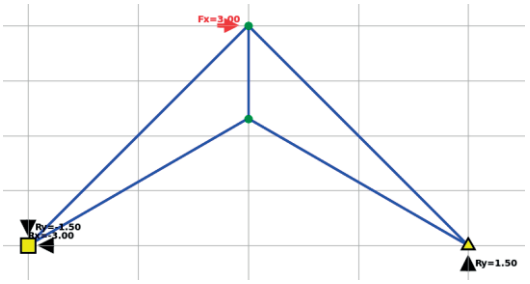

Figura 3: reacciones

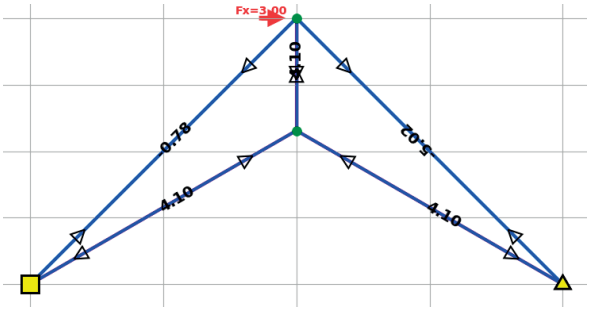

Figura 4: Fuerzas internas

RESULTADOS

La implementación computacional del método de rigidez en Python permitió realizar el análisis estructural completo de una armadura plana 2D, desde la lectura automatizada de datos hasta la generación de gráficos interpretativos.

El modelo de prueba corresponde a la armadura mostrada en la Figura 1, cuyo conjunto de datos se definió en el archivo datos2D_ejemplo_1.txt, que contiene la

información geométrica, propiedades de los elementos, condiciones de apoyo, cargas nodales y opciones de visualización.

Durante la etapa de Preproceso, el programa identificó correctamente la conectividad entre los nodos, determinando las longitudes y ángulos de cada barra a partir de las coordenadas ingresadas. Los grados de libertad se asignaron automáticamente para cada nodo, tal como se muestra en la Figura 2, garantizando la consistencia entre el modelo físico y su representación matricial.

En la fase de Proceso, el programa ensambló las matrices de rigidez locales y globales, resolviendo el sistema lineal de ecuaciones para los desplazamientos nodales. A partir de ellos, se calcularon las reacciones en los apoyos y las fuerzas internas en cada barra.

Los resultados obtenidos muestran desplazamientos coherentes con el comportamiento estructural esperado, donde el nodo cargado presenta el mayor desplazamiento vertical. Asimismo, las reacciones obtenidas satisfacen las condiciones de equilibrio global del sistema.

En la Tabla de resultados numéricos, se observa que las barras 1 y 2 están sometidas a esfuerzos de tracción (valores positivos), mientras que las barras 3 y 4 presentan compresión (valores negativos), verificando la correcta transferencia de cargas dentro de la estructura. La Figura 3 representa la deformada amplificada del sistema, evidenciando un comportamiento estructural simétrico y estable.

Finalmente, en la etapa de Postproceso, se generaron automáticamente los gráficos de la estructura original, los apoyos, las cargas, las reacciones y las fuerzas internas. El programa implementa un escalado automático de textos, símbolos y desplazamientos, lo que asegura una representación visual clara independientemente de la magnitud geométrica del modelo.

Los resultados finales confirman la exactitud y estabilidad numérica del método implementado, con matrices simétricas y definidas positivas, características inherentes al método de rigidez.

CONCLUSIONES

El desarrollo del código en Python basado en el método de rigidez demostró ser una herramienta eficaz para el análisis matricial de estructuras 2D, combinando rigor teórico con flexibilidad computacional.

Eficiencia y precisión:

El programa resolvió correctamente el sistema estructural propuesto, obteniendo resultados consistentes con los ejemplos teóricos de referencia. El uso de operaciones matriciales mediante NumPy permitió un procesamiento rápido y estable incluso para modelos con múltiples elementos.

Automatización del flujo de trabajo:

La división en tres fases —Preproceso, Proceso y Postproceso— permite un flujo estructurado y reproducible. La lectura directa del archivo de datos facilita la modificación y reutilización de modelos sin intervención manual en el código.

Visualización avanzada:

La incorporación de Matplotlib permitió representar gráficamente los grados de libertad, desplazamientos, reacciones y fuerzas internas, integrando elementos de interpretación visual que complementan el análisis numérico.

Aplicabilidad académica y profesional:

La estructura del código es fácilmente ampliable para incluir efectos térmicos, cargas dinámicas o elementos tipo viga, convirtiéndose en una herramienta didáctica ideal para cursos de análisis estructural y programación científica.

En conclusión, el análisis matricial de estructuras 2D mediante el método de rigidez

en Python constituye una plataforma potente, transparente y reproducible que une la teoría estructural con la simulación computacional, contribuyendo al fortalecimiento del aprendizaje y la investigación en ingeniería civil.

REFERENCIAS BIBLIOGRÁFICAS

Hibbeler, R. C. (2017). Estática: Mecánica para ingenieros (14.ª ed.). Pearson Educación.

Oñate, E. (2009). Cálculo de estructuras por el método de los elementos finitos: Análisis estático lineal (2.ª ed.). CIMNE.

Ortiz Soto, D. (2018). Análisis de estructuras: Problemas resueltos. Universidad Nacional de Ingeniería (UNI), Perú.

Aguiar, R. (2014). Análisis matricial de estructuras (4.a ed.). Escuela Politécnica Nacional.

McCormac, J. C. (2012a). Análisis de estructuras: Método clásico y matricial (4.a ed., 1.ª parte). Pearson Educación.

McCormac, J. C. (2012b). Análisis de estructuras: Método clásico y matricial (4.a ed., 2.ª parte). Pearson Educación.

Hurtado, J. E. (2015). Análisis matricial de estructuras con MATLAB. Editorial Alfaomega