Optimización del Cálculo de Estructuras mediante Modelos de Orden Reducido y Aprendizaje Automático No Intrusivo.

Autor: Matías Zieleniec

Supervisión académica: Franz Chouly

Supervisión industrial: Francisco Pons

Afiliación: Ingenium - Ingeniería Estructural

Institución Académica: Centro de Matemática, FCIEN-UdelaR

Fecha: 16 de Junio de 2025

Optimización del Cálculo de Estructuras mediante Modelos de Orden Reducido y Aprendizaje Automático No Intrusivo.

Resumen

Esta investigación se centra en la optimización temporal del cálculo de estructuras mediante el método de los elementos finitos (FEM). Se aborda el problema del alto costo computacional de las simulaciones de alta fidelidad, especialmente en análisis paramétricos , empleando Métodos de Orden Reducido (ROM) no intrusivos. Estos métodos son ventajosos, ya que no requieren acceso a las ecuaciones diferenciales gobernantes, sino únicamente a las soluciones (snapshots) generadas por softwares comerciales. Se proponen y evalúan dos enfoques basados en aprendizaje automático. El primer método utiliza una red neuronal para predecir los coeficientes de una base reducida, obtenida mediante Descomposición Ortogonal Propia (POD), que aproxima la solución del sistema. El segundo enfoque se centra en predecir la inversa de la matriz de rigidez reducida utilizando métodos de ensamble basados en árboles (Random Forest y Gradient Boosting). Se presentan resultados para dos casos de estudio: un bloque elástico y una losa de Kirchhoff-Love, evaluando la efectividad y precisión de cada metodología. Los hallazgos indican que ambos enfoques son prometedores, aunque su precisión varía según la complejidad del problema estructural y el método de aprendizaje automático empleado.

Palabras Clave: Método de los Elementos Finitos (FEM), Modelos de Orden Reducido (ROM), Método de la Base Reducida, Aprendizaje Automático, No Intrusivo, Descomposición Ortogonal Propia (POD), Redes Neuronales, Gradient Boosting, Random Forest, Ingeniería Estructural.

1. Introducción

Este proyecto se desarrolló en el marco de una pasantía en Ingenium - Ingeniería Estructural para la carrera de Lic. en Matemática de FCIEN-UdelaR.

El método de los elementos finitos (FEM), cuyas bases teóricas se sentaron a mediados del siglo XX y fue popularizado con la llegada de las computadoras digitales, es hoy una herramienta numérica fundamental para la simulación del comportamiento de estructuras en ingeniería. A pesar de su amplio uso y madurez, los modelos FEM detallados pueden ser computacionalmente prohibitivos,

particularmente en aplicaciones que requieren múltiples evaluaciones, como los análisis paramétricos o los procesos de optimización.

Para mitigar este costo, han surgido los Métodos de Orden Reducido (ROMs). Estas técnicas, que buscan disminuir drásticamente la dimensionalidad del problema original manteniendo una precisión aceptable, a menudo se basan en la construcción de una base reducida a partir de soluciones precalculadas. Uno de los pilares para este fin es la Descomposición Ortogonal Propia (POD), un método con raíces en el análisis de componentes principales que fue introducido en el campo de la mecánica de fluidos por Lumley en 1967 para identificar estructuras coherentes en flujos turbulentos, y que desde entonces se ha adaptado a una vasta gama de problemas físicos.

Inicialmente, la aplicación de estos métodos se centró en enfoques *intrusivos*, que requieren el conocimiento explícito de las ecuaciones diferenciales y sus formulaciones débiles. Si bien estos métodos funcionan para geometrías simples, su aplicabilidad se ve limitada en escenarios con combinación de distintos elementos estructurales o condiciones de borde complejas.

Dadas estas limitaciones, el enfoque de la investigación ha virado en la última década hacia los **métodos no intrusivos**, un cambio impulsado en gran medida por los avances en el aprendizaje automático (*Machine Learning*). Estos métodos construyen el modelo reducido directamente a partir de datos generados por el modelo de alta fidelidad (FEM), sin necesidad de acceder o modificar las ecuaciones subyacentes del sistema. Esto es particularmente relevante, ya que en la mayoría de los softwares de simulación comerciales el usuario solo tiene acceso a las salidas (soluciones o *snapshots*) y no a las matrices internas del sistema.

En este trabajo se proponen y comparan dos metodologías no intrusivas:

1. Un enfoque que utiliza la Descomposición Ortogonal Propia (POD) para construir una base reducida a partir de *snapshots* y entrena una red neuronal para mapear los parámetros de entrada del problema a los coeficientes de dicha base. La fundamentación teórica para este mapeo se apoya en el Teorema de Aproximación Universal (Cybenko, 1989, 1).
2. Un segundo enfoque que busca predecir directamente la inversa de la matriz de rigidez reducida. Para ello, se aplica POD a un conjunto de estas matrices inversas y se entrena un método de ensamble basados en árboles para predecir sus coeficientes a partir de los parámetros del problema.

2. Metodología

El procedimiento se divide en una etapa *offline*, computacionalmente intensiva, y una etapa *online*, de evaluación rápida. Se detallan a continuación los dos métodos propuestos.

2.1 Método 1: Aproximación de la Solución con Redes Neuronales

2.1.1 Etapa Offline

1. **Escoger el espacio de parámetros:** Lo primero a definir debe ser cuáles son los parámetros que vamos a hacer variar y en qué rangos (cuanto menor sea el rango mejor será la aproximación, y cuanto más parámetros hayan va a haber que aumentar el conjunto de entrenamiento debido a la maldición de la dimensionalidad).
2. **Generación de Snapshots:** Se utiliza un software de elementos finitos (FEniCS en esta investigación) para calcular un conjunto de soluciones de alta fidelidad (*snapshots*) para diferentes configuraciones paramétricas del problema estructural. Estos *snapshots* forman el conjunto de entrenamiento y testeo.
3. **Construcción de la Base Reducida (POD):** Se aplica la Descomposición Ortogonal Propia (POD) al conjunto de *snapshots* de entrenamiento utilizando la biblioteca UQpy. La POD permite identificar los modos energéticamente más significativos, que conformarán la base reducida. La elección del número de modos en la base es crucial, ya que un mayor número de modos puede dificultar la aproximación por parte de la red neuronal. Para esta elección puede ser útil conocer los valores singulares de la matriz de snapshots.
4. **Cálculo de Coeficientes de Referencia:** Con la base reducida obtenida, se calculan los coeficientes óptimos (salidas esperadas para la red) para cada *snapshot* del conjunto de entrenamiento mediante una proyección de mínimos cuadrados. Esta proyección asegura la mejor aproximación posible de las soluciones de alta fidelidad en el subespacio definido por la base reducida.
5. **Entrenamiento de la Red Neuronal:** Se entrena una red neuronal para aprender la función que mapea los parámetros de entrada del problema a los coeficientes de la base reducida calculados en el paso anterior. El objetivo es que la red pueda predecir estos coeficientes para nuevas combinaciones de parámetros. Se exploran diferentes arquitecturas de red, variando el número de capas ocultas, neuronas por capa y funciones de activación.

2.1.2 Etapa Online

En la etapa *online*, dado un nuevo conjunto de parámetros de entrada para el cual se desea conocer la solución, se utiliza la red neuronal entrenada para predecir los coeficientes de la base reducida. La solución aproximada se reconstruye entonces como una combinación lineal de los vectores de la base reducida utilizando los coeficientes predichos. Esta etapa es computacionalmente muy eficiente, ya que solo implica la evaluación de la red neuronal entrenada.

2.2 Método 2: Predicción de la Matriz de Rigidez Inversa con métodos de ensamble basados en árboles

2.2.1 Etapa Offline

1. **Escoger el espacio de parámetros:** Lo primero a definir debe ser cuáles son los parámetros que vamos a hacer variar y en qué rangos (cuanto menor sea el rango mejor será la aproximación, y cuanto más parámetros hayan va a haber que aumentar el conjunto de entrenamiento debido a la maldición de la dimensionalidad).
2. **Generación de Datos:** Se generan los *snapshots* de la solución y se extraen las matrices de rigidez A y los vectores de carga f para cada configuración paramétrica. Estos dependen del parámetro escogido
3. **Construcción de la Base Reducida (POD):** Se aplica POD a los *snapshots* para obtener la matriz de base reducida V.
4. **Reducción de las matrices de rigidez:** se reducen las matrices de rigidez con la matriz de base reducida. Si V es la matriz de base reducida con n modos y A es la matriz de rigidez entonces $V^T AV$ es la matriz de rigidez reducida de tamaño $n \times n$
5. **POD a la inversa de la matriz de rigidez reducida:** Se invierte la matriz de rigidez reducida y se vectorizan (aplanan), almacenandolas en una matriz. A esta matriz se le aplica POD obteniendo una base reducida para expresar estas matrices
6. **Entrenamiento de métodos de ensamble basados en árboles:** Se entrena un modelo de ensamble basados en árboles (trabajamos con Random Forest y Gradient Boosting) para predecir la relación entre un parámetro y los multiplicadores de la base reducida para poder aproximar la inversa de la matriz de rigidez reducida

2.2.2 Etapa Online

En esta etapa la idea es encontrar el campo de desplazamientos para un nuevo

parámetro. El métodos de ensamble basados en árboles predice los coeficientes para reconstruir la matriz de rigidez reducida (A_r^{-1}). Luego podemos resolver el sistema reducido $u_r = A_r^{-1} f_r$. En donde f_r es el vector derecho reducido, esto quiere decir que $f_r = V^T f$ (esto es la proyección al espacio reducido). Este vector reducido se puede obtener para un nuevo parámetro o se puede entrenar un modelo para predecirlo (en general los vectores derechos varían linealmente) . Finalmente, se multiplica por la matriz de base reducida para llevarlo al tamaño original (proyectar al espacio original)

---

3. Casos de Estudio y Software

Todos los códigos fueron desarrollados en Python. Las bibliotecas principales utilizadas incluyen:

- **FEniCS** (versión del 2019, no FEniCSx) para la simulación por elementos finitos y generación de *snapshots*.[1]
- **UQpy** para la implementación de POD.
- **Pytorch** para el desarrollo y entrenamiento de las redes neuronales.
- **Scikit-learn (Sklearn)** para probar otros modelos de regresión.
- **LightGBM** para entrenar el gradient boosting

Se investigaron dos estructuras principales:

1. **Bloque Elástico:** Un bloque cuadrado de lado 1 metro con alrededor de 2000 nodos, empotrado en su lado izquierdo, dividido en 9 subdominios (cuadrantes). Se aplican fuerzas de tracción o compresión en tres secciones del lado derecho (ver Anexo A).
2. **Losa de Kirchhoff-Love:** Una losa cuadrada de lado 5 metros con bordes fijos, dividida en 4 cuadrantes. En cada cuadrante se aplica una fuerza distribuída (hacia arriba o hacia abajo) y se varía el espesor de la losa (ver Anexo B). Para la discretización dividimos cada lado en 17 partes iguales, generando 578 elementos finitos (triángulos).

Las primeras pruebas incluyeron la formulación débil del problema de la losa de Kirchhoff-Love y su resolución mediante FEM en FEniCS, así como pruebas iniciales con la barra de Euler en FEniCS y RBniCS. RBniCS también se utilizó para un ejemplo

[1] Para Windows la instalación de este software puede ser compleja. Se recomienda usar una wsl con linux.

de bloque elástico con rigidez variable y para la losa de Kirchhoff-Love con bases reducidas. Sin embargo, debido a la complejidad de trabajar con las ecuaciones diferenciales, se optó por la alternativa no intrusiva basada en POD y redes neuronales.

---

4. Resultados

Analizamos los resultados para ambas metodologías

4.1 Resultados del Método 1 (Redes Neuronales)

La efectividad del método no intrusivo propuesto se evaluó en los dos casos de estudio.

4.1.1 Bloque Elástico

Para el caso de estudio del bloque elástico, se utilizó un conjunto de datos con 1000 muestras para entrenamiento y 30 para testeo. A partir de los *snapshots* de entrenamiento, se construyó una base reducida de 10 elementos mediante POD. El modelo de predicción consistió en una red neuronal de 3 capas ocultas, cada una con 48 neuronas.

La precisión del método se evaluó utilizando dos indicadores de error:

1. **Error relativo:** Mide la discrepancia global entre la solución del modelo reducido y la del modelo de alta fidelidad, calculada como la norma de la diferencia entre ambas, normalizada por la norma de la solución real. Este error cuantifica qué tan alejada está la solución predicha en su conjunto.

2. **Error relativo al desplazamiento máximo:** Se enfoca en el error local máximo. Se calcula como la mayor diferencia absoluta entre la solución predicha y la real en cualquier grado de libertad, normalizada por el valor máximo absoluto del desplazamiento real. Este indicador permite identificar la magnitud del peor error puntual en la predicción.

$$e_{relativo} = \frac{||u_{MEF}(\mu) - u_{ROM}(\mu)||}{||u_{MEF}(\mu)||}$$

$$e_{rel\,máx} = \frac{max_i(|u_{MEF}(\mu)[i] - u_{ROM}(\mu)[i]|)}{max_i(|u_{MEF}(\mu)[i]|)}$$

Los resultados en términos de error relativo y relativo a la norma del desplazamiento máximo se muestran en la Tabla 1.

Tabla 1: Resultados del modelo de red neuronal para el bloque elástico.

	Error Relativo		Error relativo al desplazamiento máximo	
	Entrenamiento	Testeo	Entrenamiento	Testeo
Promedio	0.49%	1.37%	0.55%	1.57%
Varianza	3.34e-5	5.23e-5	2.98e-5	5.86e-5
Máximo	4.14%	3.34%	8.90%	3.16%

Esta aproximación se considera muy buena con ambos métodos de evaluación del error.

Adicionalmente, se probó con regresión polinómica. Un polinomio de grado 3 generó los errores relativos mostrados en la Tabla 2.

Tabla 2: Resultados de la regresión polinómica (grado 3) para el bloque elástico. Solo con el error relativo usual

	Entrenamiento	**Testeo**
Error relativo promedio	0.5451%	0.48683%
Error relativo máximo	1.8261%	1.08789%

La regresión polinómica también genera una muy buena aproximación para este caso.

4.1.2 Losa

Para el caso de la losa, se utilizaron 4000 datos de entrenamiento y 40 de testeo. La base reducida constó de 4 elementos y la red neuronal tuvo 3 capas y 150 neuronas por capa. Los resultados se presentan en la Tabla 3.

Tabla 3: Resultados del modelo de red neuronal para la losa.

	Error Relativo		Error relativo al desplazamiento máximo	
	Entrenamiento	Testeo	Entrenamiento	Testeo
Promedio	13.23%	26.39%	14.31%	27.91%
Varianza	0.0259	0.0746	0.0283	0.0859
Máximo	122.43%	133.16%	124.70%	141.55%

Si se considera un error esperado del orden del 5%, ya que este puede ser la diferencia de dos mallados distintos en el método de los elementos finitos, según el análisis, la aproximación para la losa no fue tan buena como la esperada. También se probaron otras alternativas a redes como random forest y otros modelos lineales[2] pero estos no funcionaron.

4.2 Resultados del Método 2 (Métodos de ensamble basados en árboles)

Para este método se probó con el bloque elástico en 2 situaciones distintas. La primera como el problema anterior y la segunda es el bloque elástico dividido en 9 subdominios en donde se varía la rigidez de cada subdominio individualmente. Esto hace que en total tengamos un espacio de parámetros de dimensión 12 (3 fuerzas y 9 de rigidez).

[2] Se probó con SVR y agregar a la regresión polinómica términos como x^{-1}, x^{-2} y x^{-3}

No se puede testear este método con la losa ya que para poder formular el problema de la losa hubo que utilizar la formulación mixta modificando la matriz de rigidez. También no se predijo el valor de f, se extrajo con los snapshots.

4.2.1 Resultados para bloque elástico (4 parámetros)

Utilizando **Random Forest** con un conjunto de entrenamiento de 100 datos y un conjunto de testeo de largo 25 obtenemos en el conjunto de testeo:

Error relativo Promedio	1.4184%
Error relativo Máximo	6.4480%

Esta aproximación la consideramos muy buena.

Utilizando **Gradient Boosting** precisamos una cantidad mayor de datos para entrenar el modelo. Con 1000 datos de entrenamiento y un conjunto de testeo de largo 36 obtenemos:

Error relativo Promedio	1.2630%
Error relativo Máximo	15.7146%

Si bien esta aproximación no es tan buena como la anterior en promedio funciona bien.

4.2.2 Resultados para el bloque elástico complejizado (12 parámetros)

Utilizando **Random Forest**, con un conjunto de 1000 datos y un conjunto de entrenamiento de testeo de largo 25 se obtiene:

Error relativo Promedio	19.7734%
Error relativo Máximo	36.4412%

Esta aproximación no tiene la precisión que buscamos.

Utilizando **Gradient Boosting**, con un conjunto de entrenamiento de 1000 datos y la estructura con 2000 árboles se logra en el conjunto de testeo

Error relativo Promedio	6.3382%
Error relativo Máximo	12.1351%

Si bien no es una aproximación mala, no tiene la precisión que buscamos. Así que aumentamos el tamaño del conjunto de entrenamiento a 4000 datos, un conjunto de testeo de 40 datos y con 1000 árboles obtenemos en el conjunto de testeo:

Error relativo Promedio	4.7392%
Error relativo Máximo	15.9427%

4.3 Análisis temporal

El objetivo de esta sección es estudiar qué tanto mejoran los tiempos los distintos métodos comparados con el método de los elementos finitos original.

4.3.1 Método de los elementos finitos

Objeto	Tiempo de respuesta promedio para un parámetro
Losa (5 parámetros)	2.35 s
Bloque elástico (4 parámetros)	4.75 s
Bloque elástico (12 parámetros)	4.82 s

4.3.2 Método 1: Redes neuronales

Para el bloque elástico con 4 parámetros

Etapa	Duración en segundos
Obtención de 1000 datos de entrenamiento y POD	771.29
Entrenamiento de la red neuronal	144.07
Etapa online	0.00054

Total etapa offline: 915,36 s (15,256 minutos)

Para la losa con 5 parámetros

Etapa	Duración en segundos
Obtención de 1000 datos de entrenamiento y POD	358.50
Entrenamiento de la red neuronal	70.0
Etapa online	0.00039

Total etapa offline: 428,5 s (7,141 minutos)

Ahora si el entrenamiento de este método lo realizamos con regresión polinómica en lugar de utilizar redes neuronales para el bloque elástico con 4 parámetros.

Etapa	Duración en segundos
Obtención de 1000 datos de entrenamiento y POD	771.29
Entrenamiento de la regresión	0.1003
Etapa online	0.00002

4.3.3 Método 2: Métodos de ensamble basados en árboles

Para el bloque elástico con 4 parámetros

Gradient Boosting

Etapa	Duración en segundos
Obtención de 1000 datos de entrenamiento y POD	781.76
Entrenamiento del Gradient Boosting	5.76
Etapa online	0.00113

Total etapa offline: 787.52 s (13.125 minutos)

Random Forest

Etapa	Duración en segundos
Obtención de 1000 datos de entrenamiento y POD	781.76
Entrenamiento del Random Forest	122.15
Etapa online	0.1159

Total etapa offline: 903.91 s (15.065 minutos)

Para el bloque elástico con 12 parámetros

Gradient Boosting

Etapa	Duración en segundos

Obtención de 1000 datos de entrenamiento y POD	816.79
Entrenamiento del Gradient Boosting	34.44
Etapa online	0.0509

Total etapa offline: 851,23 s (14,187 minutos)

Random Forest

Etapa	Duración en segundos
Obtención de 1000 datos de entrenamiento y POD	816.79
Entrenamiento del Random Forest	56.37
Etapa online	0.1073

Total etapa offline: 873.16 s (14.553 minutos)

4.3.4 Comparación de Tiempos de Evaluación (Online vs. MEF)

A continuación, se comparan los tiempos de respuesta para obtener la solución de un nuevo parámetro utilizando el modelo de alta fidelidad (MEF) y los diferentes modelos reducidos en su etapa online.

Caso de Estudio	**Método**	**Tiempo de Evaluación (s)**	**Factor de Aceleración (vs. MEF)***
Bloque elástico (4 parámetros)	**MEF (Alta Fidelidad)**	**4.75**	**1x**
	Red Neuronal	0.00054	~ 8,800x

	Regresión Polinómica	0.00002	~ 237,500x
	Gradient Boosting	0.00113	~ 4,200x
	Random Forest	0.1159	~ 41x
Bloque elástico (12 parámetros)	**MEF (Alta Fidelidad)**	**4.82**	**1x**
	Gradient Boosting	0.0509	~ 95x
	Random Forest	0.1073	~ 45x
Losa (5 parámetros)	**MEF (Alta Fidelidad)**	**2.35**	**1x**
	Red Neuronal	0.00039	~ 6,000x

---

5. Discusión General

El análisis temporal (Sección 4.3) cuantifica esta ventaja de manera explícita: la etapa online presenta una **aceleración de hasta cuatro órdenes de magnitud** (e.g., 4.75 s para FEM vs. 0.00054 s para la red neuronal en el bloque elástico), validando el

enorme potencial de estos métodos para aplicaciones en tiempo real. Este beneficio justifica la inversión computacional de la etapa offline, que en los casos estudiados se mantuvo en el orden de los 15 minutos. Es notable también la diferencia de eficiencia entre los propios modelos de ML. Mientras la red neuronal ofrece la evaluación online más rápida, la regresión polinómica demuestra ser una alternativa extremadamente eficiente tanto en entrenamiento (0.10 s) como en evaluación (0.00002 s) para problemas más simples. Entre los métodos de ensamble, Gradient Boosting no solo fue más rápido de entrenar que Random Forest en estos casos, sino que su etapa online también fue significativamente más veloz.

Sin embargo, existen desventajas. Para las redes neuronales, encontrar la arquitectura óptima no es trivial y debe hacerse para cada problema. Para los métodos de ensamble basados en árboles, aunque parece ser más robusto en cuanto a la arquitectura y requerir menos datos , su etapa *online* es ligeramente más costosa por las operaciones matriciales adicionales. Además, la precisión de este segundo método puede depender de la capacidad para predecir el vector de carga reducido, si este no se puede extraer directamente.

Una limitación común a ambos enfoques es que no se puede cambiar la geometría del problema sin repetir toda la costosa etapa *offline*.

6. Conclusión

Esta investigación ha demostrado la viabilidad de utilizar métodos no intrusivos basados en aprendizaje automático para optimizar el cálculo de estructuras. Se compararon dos enfoques: uno basado en redes neuronales para predecir los coeficientes de la solución y otro basado en métodos de ensamble basados en árboles para predecir la inversa de la matriz de rigidez reducida.

Un hallazgo clave de esta investigación es la cuantificación del ahorro computacional: los modelos reducidos lograron acelerar las predicciones en varios órdenes de magnitud en comparación con el FEM, con una etapa online que se ejecuta en milisegundos frente a los segundos requeridos por la simulación de alta fidelidad. Este resultado confirma que la inversión de tiempo en una etapa offline de preparación es altamente rentable para análisis paramétricos masivos.

Los resultados muestran que ambos métodos pueden alcanzar una alta precisión, como se vio en el caso del bloque elástico. Sin embargo, su rendimiento puede degradarse en problemas más complejos o con datos insuficientes, como ocurrió con la red neuronal en el caso de la losa. El método de Gradient Boosting parece ser más

generalizable y requerir menos datos, pero a costa de una etapa *online* ligeramente más lenta.

El enfoque no intrusivo es una alternativa prometedora, especialmente cuando se trabaja con software de "caja negra". Su eficiencia en la etapa *online* lo hace atractivo para tareas de optimización y análisis paramétrico. Futuras líneas de investigación podrían enfocarse en mejorar la robustez de los métodos, explorar arquitecturas de aprendizaje profundo más avanzadas y desarrollar técnicas para manejar cambios en la geometría del problema de manera más eficiente.

Agradecimientos

Quisiera expresar un agradecimiento especial a Franz Chouly, quien fue mi profesor orientador durante la pasantía. Asimismo, extiendo mi gratitud a Francisco Pons y Fabricio Guido, cuya contribución fue fundamental para la realización exitosa de este proyecto.

Agradezco también a Ernesto Mordecki por generar el contacto con la empresa y, con ello, hacer posible esta pasantía. Por último, un reconocimiento a Gianluigi Rozza, cuyo trabajo sirvió como una importante fuente de inspiración.

Sin el valioso aporte de todos ellos, este proyecto no habría sido posible.

Referencias

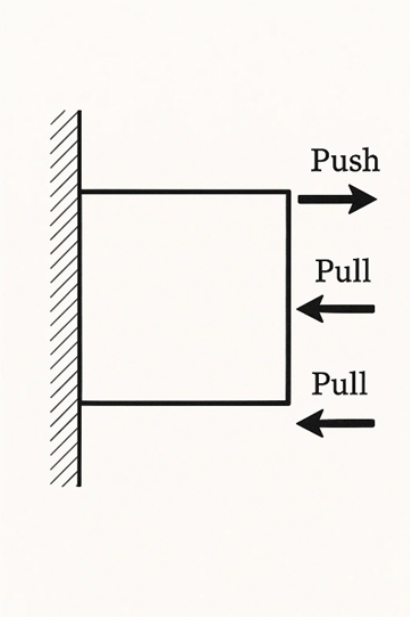
- "Material de Apoyo de unidad curricular Resistencia de Materiales 2" FING
- Langtangen, H. P., & Logg, A. (2017). *Solving PDEs in Python: The FEniCS Tutorial I* (Vol. 1). Springer. https://doi.org/10.1007/978-3-319-52462-7
- Rozza, G., Ballarin, F., Scandurra, L., & Pichi, F. (2024). *Real Time Reduced Order Computational Mechanics: Parametric PDEs Worked Out Problems*. Springer. https://doi.org/10.1007/978-3-031-38791-4
- Tannous, M., Ghnatios, C., Fonn, E., Kvamsdal, T., & Chinesta, F. (2025). *Machine Learning (ML) based Reduced Order Modelling (ROM) for linear and non-linear solid and structural mechanics* (arXiv:2504.06860v1). arXiv. https://arxiv.org/abs/2504.06860
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, *2*(4), 303-314. 10.1007/BF02551274

- The FEniCS Project. (n.d.). *Mixed Poisson equation*. FEniCS Project Documentation. Recuperado el 18 de junio de 2025, de https://olddocs.fenicsproject.org/dolfin/latest/python/demos/mixed-poisson/demo_mixed-poisson.py.html
- Baratta, I. A., & Bower, F. A. L. (2023). *Biharmonic equation*. FEniCSx v0.7.2 Documentation. Recuperado el 18 de junio de 2025, de https://docs.fenicsproject.org/dolfinx/v0.7.2/python/demos/demo_biharmonic.html
- Guzman, J., & Neilan, M. (2013). *A family of non-conforming and stable finite elements for the biharmonic equation*. [Prepublicación]. Division of Applied Mathematics, Brown University. Recuperado de https://www.dam.brown.edu/people/jguzman/documents/biharmonic_sys.pdf
- Gander, W., Gander, M. J., & Kwok, F. (2014). *Scientific Computing: An Introduction using Maple and MATLAB*. Springer.
- González Olmedo, M. (2019). *Introducción al aprendizaje automático: Apuntes para el minicurso del 7mo Coloquio Uruguayo de Matemática*. Departamento de Matemática del Litoral, Universidad de la República.

Los principales softwares y bibliotecas empleados son: FEniCS, UQpy, Pytorch, Scikit-learn (Sklearn) y LightGBM.

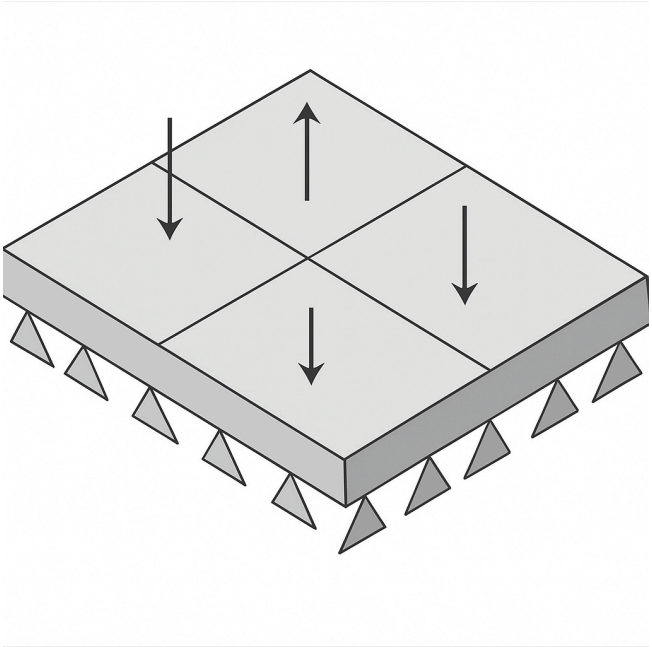
Anexos

Anexo A: Propiedades del Bloque Elástico



- Fuerzas entre -1 a 1 N
- Rigidez entre 1 y 20 Pa
- Módulo de elasticidad de 1.0 Pa
- Coeficiente de Poisson de 0.3

Anexo B: Propiedades de la Losa



- Cargas varían de -10 a 10 kN
- El espesor varía entre 4 a 60 centímetros
- Módulo de elasticidad 30 GPa
- Coeficiente de Poisson de 0.3