

UCUENCA

Universidad de Cuenca

Facultad de Ingeniería

Carrera de Ingeniería Civil

Modelización de mecanismos de falla dúctiles en barras utilizando redes neuronales artificiales


Trabajo de titulación previo a la obtención del título de Ingeniero Civil

Autor:

Omar Fernando León Iñiguez


Director:

Esteban Patricio Samaniego Alvarado

ORCID:  0000-0002-8728-491X

Codirector:

Ángel Oswaldo Vázquez Patiño

ORCID:  0000-0002-1679-3995

Cuenca, Ecuador

2023-08-07

Resumen

El presente trabajo de investigación se centra en la modelización de mecanismos de falla dúctil en una barra unidimensional. El objetivo principal es utilizar métodos basados en Physics-Informed Neural Networks (PINN) y Machine Learning, empleando el enfoque variacional, para modelizar el mecanismo de falla dúctil y la localización de deformaciones. Se desarrollaron dos implementaciones de PINN basadas en el principio variacional, utilizando diferentes ecuaciones de minimización de energía que son equivalentes entre sí. Los resultados obtenidos demuestran que las redes neuronales son capaces de capturar el comportamiento elastoplástico sin la necesidad de herramientas complejas como phase-fields. Este enfoque numérico se presenta como una opción prometedora en comparación con métodos alternativos como los elementos finitos, especialmente para problemas de dimensiones superiores, donde otros métodos muestran limitaciones. Esto abre nuevas líneas de investigación en el campo de la modelización de mecanismos de falla en sólidos. Se demostró que estas redes neuronales, aplicadas mediante el principio variacional, ofrecen una precisión suficiente en comparación con las soluciones analíticas. Como recomendación, se sugiere profundizar en la naturaleza de las redes neuronales como método para la resolución de problemas en la mecánica de sólidos, así como implementar redes neuronales en la resolución de problemas en 2D y 3D, lo cual representa una línea de investigación futura.

Palabras clave: PINN, redes neuronales, mecánica de sólidos, ductilidad, plasticidad.



El contenido de esta obra corresponde al derecho de expresión de los autores y no compromete el pensamiento institucional de la Universidad de Cuenca ni desata su responsabilidad frente a terceros. Los autores asumen la responsabilidad por la propiedad intelectual y los derechos de autor.

Repositorio Institucional: <https://dspace.ucuenca.edu.ec/>

Abstract

This research work focuses on modeling mechanisms of ductile failure in a one-dimensional bar. The main objective is to use methods based on Physics-Informed Neural Networks (PINN) and Machine Learning, employing the variational approach, to model the mechanism of ductile failure and deformation localization. Two implementations of PINN have been developed based on the variational principle, using different energy minimization equations that are equivalent to each other. The obtained results demonstrate that neural networks are capable of capturing elastoplastic behavior without the need for complex tools such as phase-fields. This numerical approach presents a promising option compared to alternative methods like finite elements, particularly for problems in higher dimensions where other methods show limitations. This opens up new lines of research in the field of modeling failure mechanisms in solids. It has been shown that these neural networks, applied through the variational principle, offer sufficient accuracy compared to analytical solutions. As a recommendation, it is suggested to further explore the nature of neural networks as a method for problem-solving in solid mechanics, as well as to implement neural networks in solving problems in 2D and 3D, which represents a future line of research.

Keywords: PINN, neural networks, mechanic of solids, ductility, plasticity.



The content of this work corresponds to the right of expression of the authors and does not compromise the institutional thinking of the University of Cuenca, nor does it release its responsibility before third parties. The authors assume responsibility for the intellectual property and copyrights.

Institutional Repository: <https://dspace.ucuenca.edu.ec/>

Índice de Contenido

1. Introducción	9
1.1. Antecedentes	9
1.2. Motivación	14
1.3. Objetivos.....	15
1.3.1. Objetivo General.....	15
1.3.2. Objetivos Específicos.....	15
2. Marco Teórico.....	16
2.1. Mecánica de Fractura	16
2.1.1. Fundamentos de la mecánica de medios continuos.....	16
2.1.2. Principios generales de la mecánica de fallo localizado.....	18
2.1.3. Enfoque variacional a la localización de deformaciones	21
2.1.4. Mecanismos de falla frágil.....	22
2.1.5. Mecanismo de falla dúctil.....	24
2.2. Métodos numéricos.....	26
2.2.1. Métodos numéricos en la ingeniería civil.....	26
2.2.2. Machine Learning	27
2.2.3. Physics Informed Neural Networks (PINN)	28
3. Materiales y Metodología	29
3.1. Materiales	29
3.2. Metodología.....	30
4. Configuración geométrica y mecánica del caso de estudio	31
4.1. Planteamiento del caso de estudio	31
4.2. Configuración geométrica y propiedades mecánicas de la barra 1D.....	32
4.3. Mecánica de la barra 1D, caso frágil.....	32
4.4. Mecánica de la barra 1D, caso dúctil	33
5. Modelización de la barra 1D	34

UCUENCA

	5
5.1. Solución analítica del problema unidimensional.....	34
5.2. Implementación de la PINN	37
5.3. Modelización de la barra 1D, caso frágil	40
5.4. Modelización de la barra 1D, caso dúctil.....	44
6. Discusión y Conclusiones	53

Índice de Ilustraciones

Ilustración 1.- Evolución de la perspectiva de la fractura (Cotterell, 2002).....	10
Ilustración 2.- El esqueleto de acero de la oficina y almacén de Spicer Brother's, New Bridge Street (Clarke, 2014).....	11
Ilustración 3.- Esquema de los tres modos fundamentales de fractura (Ritchie & Liu, 2021)13	
Ilustración 4.- Tipos de comportamiento elastoplástico	20
Ilustración 5.- Localización unidimensional	21
Ilustración 6.- Composición de la energía total de un sólido como el área bajo la curva esfuerzo-deformación. En rojo la energía disipada y en azul la energía almacenada	22
Ilustración 7.- Potencial de energía almacenada en el caso de fractura frágil	23
Ilustración 8.- Curva esfuerzo-deformación del modelo constitutivo con ablandamiento	24
Ilustración 9.- Descomposición de la energía en almacenada (azul) y disipada (rojo) para el modelo con ablandamiento y sin pérdida de rigidez.....	25
Ilustración 10.- Esquema de red neuronal (<i>Colored Neural Network</i> , 2019).....	28
Ilustración 11.- Arquitectura de la NN regular (izquierda) en combinación con PINN (derecha).	29
Ilustración 12.- Definición del problema unidimensional	31
Ilustración 13.- Esquema del desplazamiento en la barra producto de la localización.....	35
Ilustración 14.- Desplazamiento de la barra, solución analítica	35
Ilustración 15.- Deformación de la barra, solución analítica.....	36
Ilustración 16.- Tensión en la barra, solución analítica	36
Ilustración 17.- Desplazamiento para distintos casos de desplazamiento en el extremo	42
Ilustración 18.-Deformación para distintos casos de desplazamiento en el extremo	43
Ilustración 19.- Tensión para distintos casos de desplazamiento inicial en el extremo.....	43
Ilustración 20.- Resultado de desplazamiento para caso dúctil ($d=5.4$).....	45
Ilustración 21.- Resultado de deformación para caso dúctil ($d=5.4$)	46
Ilustración 22.- Resultado de tensión para caso dúctil ($d=5.4$)	46
Ilustración 23.- Resultados de desplazamiento, caso dúctil, para distintos desplazamientos en el extremo	47
Ilustración 24.- Resultados de deformación, caso dúctil, para distintos desplazamientos en el extremo.....	47
Ilustración 25.- Resultados de tensión, caso dúctil, para distintos desplazamientos en el extremo.....	48
Ilustración 26.- Descomposición del desplazamiento. En rojo la componente regular y en azul la componente del salto.	49

Ilustración 27.- Desplazamiento y sus componentes regular y salto para distintos desplazamientos en el extremo.....	51
Ilustración 28.- Resultado de desplazamiento caso dúctil.....	52
Ilustración 29.- Resultado de deformación caso dúctil.....	52
Ilustración 30.- Resultado de tensión caso dúctil.....	53

Agradecimientos

A mi familia,

Quiero expresar mi profundo agradecimiento por haber sido fundamentales durante mi formación. Su constante presencia, amor y apoyo han sido un motor imprescindible para alcanzar este importante logro en mi vida.

En particular a mi madre, gracias por creer en mí y por ser mi fuente inagotable de motivación, por alentarme en los momentos de duda y por celebrar cada pequeño avance y éxito a lo largo de este camino. Su comprensión y paciencia durante las largas horas de estudio y dedicación han sido invaluable.

A mis amigos,

Gracias por estar ahí en los momentos de estrés y presión, por ser aliados en las largas noches de estudio y por recordarme la importancia de equilibrar el trabajo académico con momentos de diversión y descanso.

A lo largo de este camino, hemos compartido risas, desafíos y momentos inolvidables juntos. Han sido compañeros de estudio, confidentes y motivadores constantes. Su presencia ha hecho que cada obstáculo sea más llevadero y cada triunfo sea más significativo.

A mis maestros,

Deseo expresar mi más sincero agradecimiento a todos aquellos maestros que han dejado una huella profunda en mi formación académica y personal. Su dedicación y pasión por la enseñanza han sido una inspiración constante y han influido de manera significativa en mi crecimiento y éxito.

En particular deseo agradecer a dos personas excepcionales: el Ing. Esteban Samaniego y el Ing. Ángel Vázquez. Su experiencia, conocimiento y mentoría han sido fundamentales en mi formación y en el éxito de esta tesis. Valoramos enormemente su guía experta y su compromiso con la excelencia académica.

1. Introducción

1.1. Antecedentes

La ingeniería civil es una rama del conocimiento que ha acompañado a la humanidad desde los albores de su existencia como sociedad civilizada, como tal se dedica fundamentalmente al diseño, construcción y mantenimiento de infraestructuras. Esta definición permite que el ingeniero civil se desenvuelva en una amplia rama de subdisciplinas, por ejemplo, ingeniería estructural, ingeniería geotécnica, ingeniería hidráulica o ingeniería sísmica. (Kosky et al., 2021).

Atendiendo a esto, la ingeniería civil puede ser dividida en dos grandes áreas de estudio en lo que a fenómenos físicos se refiere: la mecánica de sólidos y la mecánica de fluidos. La mecánica de fluidos es la rama de la mecánica dedicada al estudio de fluidos en movimiento o en reposo, y a las fuerzas particulares que se producen dentro de estos sistemas (Rubenstein et al., 2022). Por otro lado, la mecánica de sólidos es la rama de las ciencias físicas que se ocupa de la deformación y el movimiento de los sólidos bajo la acción de cargas externas, ya sean estas fuerzas, desplazamientos, aceleraciones o la combinación de estas, que dan lugar a la aparición de fuerzas internas (Sancaktar, 2019). Esta rama de la ciencia es de fundamental importancia en la ingeniería civil, especialmente en las áreas específicas de ingeniería estructural, ingeniería sísmica y ciencia de materiales.

Tanto la mecánica de sólidos y la de fluidos comparten una característica fundamental y es que sus fenómenos son modelados usando Ecuaciones en Derivadas Parciales (EDPs). Las EDPs constituyen el componente central de una gran variedad de sistemas físicos, y su resolución es esencial para el diseño óptimo de una gran variedad de estructuras y para el análisis de multitud de fenómenos tan variados como el transporte de sedimentos (Man & Tsai, 2007), sistemas de amortiguamiento en edificios (Chen et al., 2010) y el comportamiento mecánico de suelos (Ho & Fatahi, 2016) .

Uno de estos fenómenos, que resulta de gran interés para la ingeniería civil y la mecánica de sólidos, es la fractura, este constituye un concepto fundamental en la mecánica de sólidos, en particular a lo relacionado con ciencia de los materiales (Rossmanith, 1997). Este fenómeno es estudiado a través de la mecánica de fractura que se dedica a la interpretación juiciosa de los mecanismos de falla y de fractura (Ritchie & Liu, 2021). El desarrollo de esta ciencia se produce en épocas recientes, fundamentalmente en la segunda mitad del siglo XX, en la época de la posguerra, desarrollada con el fin de estudiar cuantitativamente el inicio de

la fractura y, de esta forma, dar paso al desarrollo de estructuras más seguras como las utilizadas en el transporte, la construcción y la producción de energía (Zehnder, 2012).

Aun así, la noción de fractura y el interés por su estudio desde una perspectiva científica han acompañado a la civilización desde épocas tempranas. Como se puede observar en la ilustración 1, personajes como Leonardo da Vinci y Galileo Galilei estudiaron la fractura y comprendieron un aspecto fundamental de esta ciencia: la escala (Cotterell, 2002).

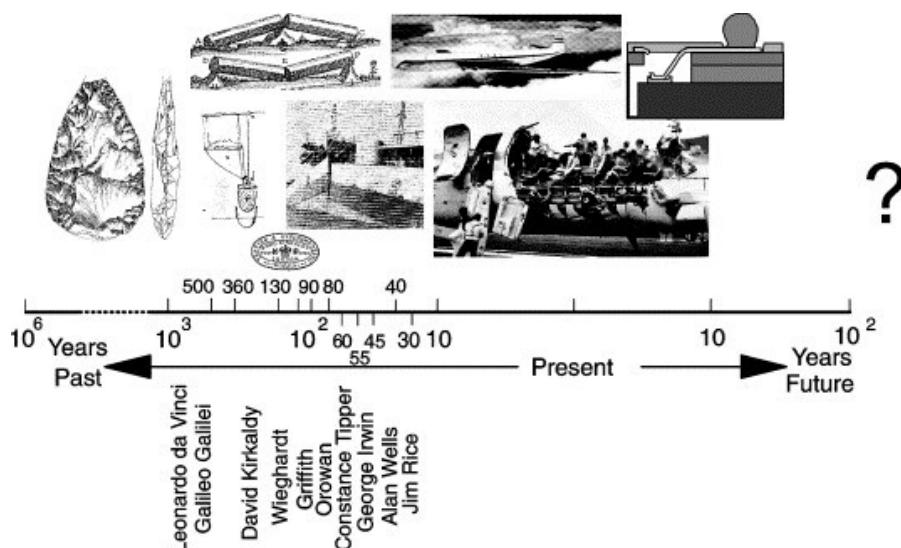


Ilustración 1.- Evolución de la perspectiva de la fractura (Cotterell, 2002)

Con la industrialización en la Inglaterra del siglo XIX y XX se comenzó a utilizar de forma masiva nuevos materiales en la construcción, particularmente el acero. Este tuvo un ‘boom’ en la época al permitir construir edificaciones e instalaciones más grandes, como la mostrada en la ilustración 2. Naturalmente, producto de este uso masivo, el interés en el estudio del comportamiento de este novedoso material creció y con él también lo hizo la necesidad de comprender de manera más precisa los mecanismos de fractura.

Esta atención dada al acero produjo que David Kirkaldy abriera su laboratorio de pruebas y trabajos experimentales en 1865, sus ensayos dentro de este condujeron a la publicación de sus experimentos (Kirkaldy, 1866) en forma de compendio. En este se describen aspectos relacionados con el comportamiento mecánico del acero y algunos problemas relacionados con el fenómeno específico de la fractura.

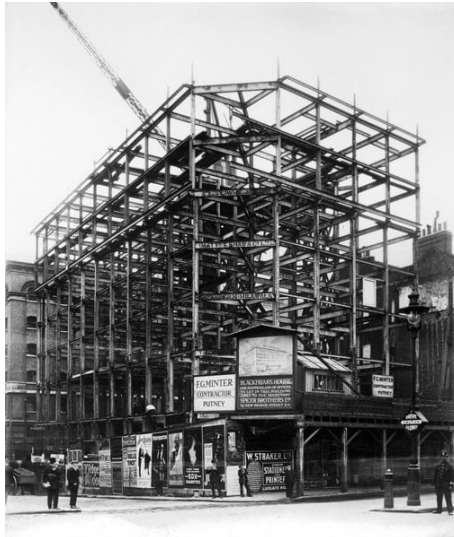


Ilustración 2.- El esqueleto de acero de la oficina y almacén de Spicer Brother's, New Bridge Street (Clarke, 2014)

Las bases teóricas de mecánica de fractura se sustentan fundamentalmente en dos autores: Griffith (Griffith, 1921, 1924) y Wieghardt (Wieghardt et al., 1995). Este último se interesó en particular en una paradoja que resulta conveniente analizar, las tensiones en la punta de una grieta delgada de un cuerpo elástico son infinitas independientemente de la magnitud del esfuerzo aplicado, siguiendo esto propuso que la ruptura no se produce cuando la tensión en un punto supera un valor crítico, sino que se produce cuando la tensión en una pequeña parte del cuerpo supera un valor crítico. Esta idea fue verdaderamente desarrollada por Griffith quien, tras una serie de experimentos, logro definir este concepto de esfuerzo crítico. La teoría de fractura de Griffith establece que *'una grieta se propagará cuando la reducción de energía potencial que se produce debido al crecimiento de la grieta sea mayor o igual que el aumento de energía superficial debido a la creación de nuevas superficies libres'* (Zehnder, 2013).

Griffith desarrolló su teoría fundamentalmente alrededor de los casos de fractura frágil, de hecho, los experimentos que realizó para validar sus propuestas (Griffith, 1921) fueron hechos sobre vidrio, un material cuyo comportamiento frágil es evidente. Esta limitación impidió que su trabajo fuese ampliamente difundido. No fue hasta las contribuciones de Irwin (Irwin, 1948) que la plasticidad fue introducida dentro de la teoría de Griffith; fue él quien concluyo que, al añadir la energía superficial a la teoría, esta ampliaba su rango de aplicabilidad y por ende tenía más interés y relevancia en el campo de la ingeniería.

Los aportes de Irwin condujeron al desarrollo de una teoría de mecánica de fractura elástico lineal (LEFM por sus siglas en inglés). Esta teoría se convirtió en uno de los conceptos más relevantes de la mecánica de medios continuos, con una gran variedad de aplicaciones desde

el crecimiento de grietas de fatiga hasta grietas por esfuerzos y corrosión. Sin embargo, el modelo elástico lineal continuaba prediciendo valores infinitos de tensión en las cercanías de las grietas producidas, esto era evidencia de que había un fallo en el centro de la teoría y que esta dejaba de ser aplicable bajo ciertas circunstancias. Esta situación motivó el desarrollo de una teoría elastoplástica para la mecánica de fractura.

Es justamente en este aspecto donde es necesario hacer una revisión de otros dos modelos fenomenológicos esenciales de la mecánica del medio continuo: la plasticidad y el daño. La plasticidad está asociada a deformaciones irreversibles que generan disipación de energía, mientras que el daño se asocia a las nucleaciones de vacío y representa la última etapa de disipación de energía antes de la fractura.

Una vez que se tiene en cuenta la plasticidad, es posible clasificar la descripción macroscópica de la fractura en tres grupos principales: frágil, dúctil y por fatiga. La fractura frágil y dúctil implican una sola aplicación de carga, mientras que la fractura por fatiga se produce por una carga cíclica. La fractura frágil y la fractura dúctil son términos bastante generales que describen los dos extremos opuestos del espectro de fractura que requieren algunas consideraciones importantes. En general, la principal diferencia entre fractura frágil y dúctil puede atribuirse a la cantidad de deformación plástica que experimenta el material antes de que se produzca la fractura. Los materiales dúctiles presentan grandes cantidades de deformación plástica, mientras que los materiales frágiles muestran poca o ninguna deformación plástica antes de la fractura.

Desde la perspectiva macroscópica también se puede decir que la fractura se caracteriza por discontinuidades superficiales que a su vez se pueden describir como discontinuidades en el campo de desplazamientos. Existen tres mecanismos fundamentales de fractura (Ritchie & Liu, 2021), que se muestran en la ilustración 3, estos son:

- Modo I: Tensión
- Modo II: Corte planar
- Modo III: Corte anti-planar

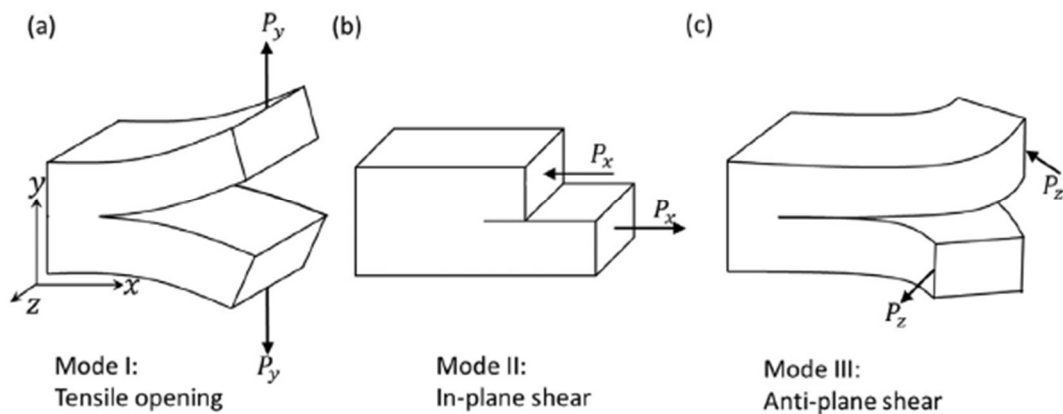


Ilustración 3.- Esquema de los tres modos fundamentales de fractura (Ritchie & Liu, 2021)

Este fenómeno puede ser estudiado en diversos materiales como metales o polímeros, y cada uno de estos presentará una serie de singularidades respectivas, aun así, pese a las diferentes estructuras físicas de los materiales, se puede considerar que presentan comportamientos mecánicos similares. En general, todos presentan un rango de comportamiento elástico seguido por un rango de fluencia (deformaciones plásticas) que da paso al daño y por último a la fractura macroscópica (Lemaitre, 1992).

En lo que respecta a la modelización del fenómeno se puede hablar de dos enfoques principales que se han utilizado a lo largo de los años: formulaciones clásicas y formulaciones variacionales. Las formulaciones clásicas se basan en la teoría de materiales estándar, donde se analiza la evolución del material de manera local (Borja, 2013; Simo & Hughes, 1998). El enfoque variacional parte de una formulación energética, misma que se encuentra en el trabajo de Francfort y Marigo (Francfort & Marigo, 1998) con un enfoque a la falla frágil; también es posible encontrar más aplicaciones de este enfoque variacional a la mecánica de fractura en el artículo de los mismos autores de 2008 (Bourdin et al., 2008).

El acoplamiento de plasticidad o daño permite que aparezcan discontinuidades en el campo de desplazamientos antes de alcanzar el estado de fractura, lo que permite que la formación de grietas en materiales dúctiles se represente adecuadamente (Rodríguez Cedillo & Ulloa Vanegas, 2015). Esta línea de investigación fue continuada (Ulloa et al., 2016) en un marco de formulación energética mediante la minimización de un funcional de energía global.

Es justamente dentro de esta línea de investigación, la del enfoque variacional, donde se desarrolla el presente trabajo, ya que dados los recientes avances en la formulación de nuevos métodos numéricos que hacen uso de Machine Learning y Deep Learning, es posible analizar la mecánica de fractura dentro del contexto de localización de deformaciones usando

como principal herramienta un funcional de energía que permita, al ser minimizado, cumplir con las condiciones tanto de contorno, como con las condiciones dadas por la naturaleza del material (ecuaciones constitutivas).

1.2. Motivación

Todo material está sometido constantemente a esfuerzos y deformaciones que generan altas demandas. Estas demandas eventualmente pueden provocar fenómenos de plasticidad, daño y, por último, fractura, generando un fallo. Este fallo se vuelve de particular interés cuando los materiales constituyentes demandados son parte de elementos o estructuras, ya que al diseñar y construir edificaciones se pretende garantizar su estabilidad y durabilidad a fin de precautelar bienes económicos y humanos.

Aun así, dado el avance en el conocimiento del comportamiento de los materiales, y con la motivación de que la infraestructura sea más rentable económicamente, se diseñan estructuras sujetas a esfuerzos cada vez más cercanos a su última resistencia. Es por esta razón que para un diseño racional se requiere un conocimiento exhaustivo de los fenómenos envueltos en el comportamiento de una estructura y sin duda la fractura es parte obligatoria de este conocimiento.

Este es el punto de partida de este trabajo, ya que, considerando la multitud de enfoques, tanto conceptuales como computacionales que se han usado para estudiar el fenómeno de fractura, y considerando que aún existen grandes vacíos de conocimiento, la búsqueda de alternativas se presenta como una línea de investigación relevante.

En el campo de los métodos numéricos se han dado avances interesantes últimamente, entre ellos el desarrollo de Physics-Informed Neural Networks (PINN) es de particular interés. Este método ya ha sido propuesto para la resolución de sistemas de EDPs (Raissi et al., 2018). Su uso se basa en técnicas que incluyen el uso de redes neuronales artificiales y métodos de ajuste de ecuaciones diferenciales, que controlan el sistema físico modelado, usando varios puntos del dominio que se denominan puntos de colocación (Katsikis et al., 2022). La aplicabilidad de las PINN a la resolución de una serie de problemas ha sido planteada exitosamente por diversos autores y desde una multitud de enfoques (Goswami et al., 2020; Samaniego et al., 2020). Estos avances indican que la aplicación de PINN en áreas de la ingeniería puede, potencialmente, resolver, entre otros, problemas de optimización y gasto computacional (Vadyala et al., 2022). Para aprovechar estos potenciales avances en diversas

áreas de la ingeniería, se requiere explorar el método PINN en múltiples aplicaciones, por ejemplo, la falla dúctil en sólidos.

El uso de las PINN ha sido presentado en la literatura haciendo uso explícito de las ecuaciones de balance y compatibilidad, es decir, utilizando la formulación fuerte del problema. Sin embargo, su uso a partir del enfoque variacional, esto es, considerando fundamentalmente la minimización de la energía como condición a cumplir, ha sido poco explorado en el estudio de la mecánica de sólidos y de la mecánica de fractura, por lo cual explorar este enfoque resulta de particular interés no solo por sus posibilidades en lo referente al método en sí mismo sino también por sus posibles aplicaciones prácticas.

En atención a lo señalado se puede deducir que ser capaces de modelar fallas dúctiles de manera fiable es la base para su simulación en un tiempo de cómputo razonable. Esto permitiría, potencialmente, a los profesionales de la ingeniería civil hacer uso de simulaciones realistas en etapas de diseño iterativo. Lo anterior se traduciría en una mayor productividad en la construcción y en elementos diseñados de manera óptima y eficiente. Adicionalmente, mejorar el modelado de los mecanismos de falla permitiría abrir nuevas líneas de investigación.

1.3. Objetivos

1.3.1. Objetivo General

El objetivo principal es modelizar el mecanismo de falla dúctil y localización de deformaciones en una barra a través del uso de métodos basados en Physics-Informed Neural Networks (PINN) y Machine Learning, empleando el enfoque variacional en el contexto de la localización de deformaciones.

1.3.2. Objetivos Específicos

- Elaborar una aproximación a la modelización matemática de mecanismos de falla dúctil incorporando principios físicos a través de las ecuaciones diferenciales de gobierno respectivas considerando un enfoque variacional para la resolución del problema.
- Implementar un modelo numérico basado en PINN para la resolución del problema de falla dúctil en una barra unidimensional.

- Comparar los resultados obtenidos a partir del método implementado con los obtenidos a través de otros métodos clásicos, en particular la solución analítica .

2. Marco Teórico

2.1. Mecánica de Fractura

2.1.1. Fundamentos de la mecánica de medios continuos

La mecánica de fractura es un fenómeno que, tal como se mencionó anteriormente, se enmarca dentro de un marco conceptual aún más grande que es la conocida mecánica de medio continuos. Para la formulación matemática de la diversidad de fenómenos físicos que puede sufrir un medio continuo se hace uso de tres componentes básicos: la cinemática, las ecuaciones de balance y las ecuaciones constitutivas. Mientras que los dos primeros son universales y se deben cumplir para cualquier sistema físico, las ecuaciones constitutivas son específicas de cada material, aunque de igual manera deben estar sujetas a principios físicos establecidos por la termodinámica.

En lo que respecta a la cinemática esta se puede definir como el estudio del movimiento de un cuerpo (o sistema de cuerpos) sin considerar directamente las fuerzas o los campos que lo afectan (Hussein, 2007), también se la puede definir como la rama de la mecánica que describe el movimiento sin describir las causas del mismo (Arfken et al., 1984). De esta forma si se considera una región de estudio Ω , y se denomina Ω_0 a su configuración de referencia en el instante de referencia t_0 , al sufrir esta región una deformación, su nueva configuración está determinada por el vector de desplazamiento $\mathbf{u}(\mathbf{x}, t)$, donde \mathbf{x} representa la posición de algún punto del dominio en el espacio y t es el tiempo. Esta nueva configuración Ω_t representa en esencia el movimiento que ha realizado el cuerpo con respecto al instante de referencia.

Partiendo de estas nociones es posible construir un modelo que relacione la deformación con el desplazamiento, esta relación se describe a través del tensor de deformaciones. Considerando la teoría de pequeñas deformaciones, aplicable al presente trabajo ya que el desplazamiento y el gradiente de desplazamiento se mantiene pequeños durante todo el análisis, se puede escribir como:

$$\epsilon = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^t) \quad (1)$$

En lo que respecta a las ecuaciones de balance es fundamental que se cumpla lo siguiente:

- Balance de masa
- Balance del momento lineal
- Balance del momento angular
- Balance de la energía
- Principio de desigualdad de la entropía

Las dos últimas garantizan el cumplimiento de las dos primeras leyes de la termodinámica, y la última en particular está relacionada con el concepto de irreversibilidad. Por su parte, la conservación de la masa dice que

$$\frac{d}{dt} \int_{\Omega} \rho(\mathbf{x}, t) dV = 0. \quad (2)$$

La ecuación anterior implica que un cuerpo continuo no puede variar su masa al sufrir una deformación. Dada la suposición de pequeñas deformaciones y a la densidad unitaria considerada, el balance de masa siempre se satisface.

Al introducir el tensor de tensiones de Cauchy σ , definido como $\sigma = \mathbb{C}:\epsilon$, o para el caso unidimensional $\sigma = E\epsilon$, el balance del momento lineal se puede escribir de forma local como

$$\nabla \sigma + \mathbf{b} - \rho \ddot{\mathbf{u}} = 0. \quad (3)$$

Donde \mathbf{b} son las fuerzas de masa, que se incluyen en el balance de la cantidad de movimiento. Del balance del momento angular se tiene que

$$\sigma = \sigma^t. \quad (4)$$

Un sistema es una cantidad de materia ocupando una posición específica en el espacio; por otro lado, su estado está determinado por una serie de variables (Alessi, 2013). Estas se conocen como variables de estado y su determinación no es simple ni única, ya que depende del sistema que se quiere analizar (Maugin & Muschik, 1994). El sistema definido en este trabajo es un cuerpo deformable unidimensional sin interacción con su entorno más allá del desplazamiento impuesto en el extremo.

Hasta este punto, se han planteado las ecuaciones de balance en su forma tridimensional. El objetivo de este trabajo es modelar un sistema unidimensional sometido a esfuerzos de tracción, por lo que la formulación posterior se la realizara solo definiendo las funciones unidimensionales respectivas.

Una de las características más importantes que se logra representar a través del modelo de plasticidad es el desarrollo de deformaciones permanentes en el material, para lo cual, se asume la descomposición aditiva de la deformación total de la siguiente manera:

$$\epsilon = \epsilon^e + \epsilon^p \quad (5)$$

donde ϵ^e es la deformación elástica reversible y ϵ^p es la deformación plástica permanente.

2.1.2. Principios generales de la mecánica de fallo localizado

En mecánica de sólidos, uno de los fenómenos cuya descripción es especialmente compleja es la llamada localización de deformaciones, la cual se produce cuando estas se concentran en regiones restringidas de un sólido (bandas de localización). Este tipo de comportamiento suele estar relacionado con la utilización de leyes constitutivas con ablandamiento. La localización de deformaciones es un fenómeno de carácter general que se asocia con diferentes tipos de fallo específicos, por ejemplo, fractura en materiales frágiles y cuasi frágiles o bandas de corte en geo materiales y en metales.

El ablandamiento, tal como se mencionó, conduce a la localización de deformaciones, que, en el caso de materiales cuasi frágiles, se puede asimilar a la llamada zona de proceso de la fractura. En el caso de materiales plásticos, esta se puede asociar con bandas de corte y superficies de deslizamiento.

Se pueden distinguir dos formas diferentes de analizar la localización de deformaciones, que a su vez pueden asociarse con diferentes tipos de fallo. Así pues, se puede asociar el fenómeno de localización con la concentración de deformaciones inelásticas en la banda en la cual se producirá una discontinuidad en el campo de deformaciones (Rice, 1976; Rudnicki & Rice, 1975). A esta superficie de discontinuidad se la llama discontinuidad débil.

Por otro lado, desde un punto de vista macroscópico, la zona de localización se puede ver como una superficie de discontinuidad del campo de desplazamiento, a este se le denomina discontinuidad fuerte (Manzoli et al., 1999).

Ambas concepciones teóricas, la discontinuidad débil y fuerte, están asociadas con diferentes modelos, estando la discontinuidad débil asociada con un modelo continuo y la discontinuidad fuerte con un modelo discreto. Para el enfoque utilizado en este trabajo se hace uso de ecuaciones constitutivas en su forma continua y por tanto se recurre al modelo de discontinuidad débil.

La localización de deformaciones se produce cuando en una zona restringida del sólido (la banda de localización) se produce una gran concentración de las deformaciones. En estricto sentido, se trata de una inestabilidad del material. La localización de las deformaciones puede producirse incluso cuando se tienen distribuciones homogéneas de las tensiones.

Los dos retos fundamentales en la descripción de la localización de deformaciones son asegurar que se disipe la cantidad correcta de energía y definir el lugar geométrico de la banda de localización. Estas tienen su reflejo en los modelos numéricos obtenidos después de usar alguna técnica de discretización como el método de los elementos finitos. La información necesaria no puede obtenerse de modelos constitutivos estándar provenientes de la mecánica de medios continuos. Como consecuencia, estrategias complementarias deben adoptarse.

En este trabajo se tratará con un sistema elastoplástico unidimensional, para este objetivo se usará las ecuaciones constitutivas presentadas en el trabajo de Simo y Hughes (1998). Primero se define una relación incremental entre la tensión y la deformación de forma bien conocida, la ley de Hooke, considerando a su vez la definición de deformación que se ha presentado anteriormente, por tanto

$$\dot{\sigma} = E(\dot{\epsilon} - \dot{\epsilon}^p) = E\dot{\epsilon}^e. \quad (6)$$

La deformación plástica se define mediante la de flujo como

$$\dot{\epsilon}^p = \gamma m \quad (7)$$

donde m indica el sentido del flujo plástico, mientras que el multiplicador plástico $\gamma \geq 0$ indica su magnitud. El dominio elástico, donde no hay evolución de deformación plástica, está dado por el espacio de tensiones

$$\phi(\sigma, \mathbf{q}) < 0 \quad (8)$$

donde \mathbf{q} representa el conjunto de variables internas que gobiernan la evolución del dominio elástico. Para el caso de endurecimiento isótropo, se define el criterio de fluencia de la forma

$$\phi(\sigma, q) = |\sigma| - \sigma_y + q \leq 0 \quad (9)$$

donde σ_y es la tensión de fluencia y $q \leq \sigma_y$ representa la variable de endurecimiento/ablandamiento dada por

$$q = -H(q)\alpha \quad (10)$$

donde α representa la deformación plástica equivalente $\alpha = \gamma s$ y H es el módulo de endurecimiento/ablandamiento. Para los objetivos del presente trabajo se puede considerar el caso particular de plasticidad asociada que conduce a

$$m = \text{sign}(\sigma)$$

$$s = 1$$

$$\alpha = \gamma$$

De esto se concluye que

$$\dot{\epsilon}^p = \gamma \text{sign}(\sigma).$$

Se clasifica el tipo de comportamiento elastoplástico según el valor de H (ilustración 4) de la siguiente manera:

$$\begin{cases} H > 0: & \text{endurecimiento} \\ H = 0: & \text{plasticidad perfecta} \\ H < 0: & \text{ablandamiento} \end{cases}$$

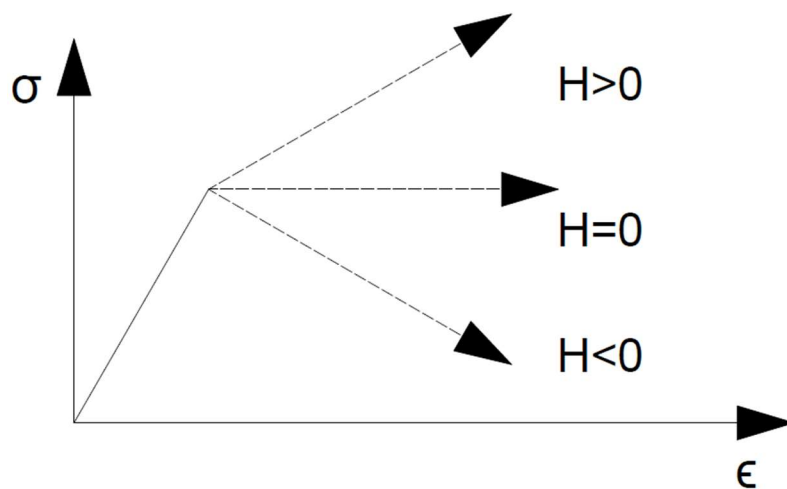


Ilustración 4.- Tipos de comportamiento elastoplástico

Los comportamientos elástico y elastoplástico quedan completamente caracterizados por las condiciones de carga y descarga de Kuhn-Tucker:

$$\gamma \geq 0 \quad \phi(\sigma, q) \leq 0 \quad \gamma \phi(\sigma, q) = 0 \quad (11)$$

Se establecen, por tanto, las condiciones de carga y descarga

$$\begin{aligned}
 \phi < 0 &\quad \Rightarrow \quad \gamma = 0 \Rightarrow \dot{\sigma} = E\dot{\epsilon} && \text{(elastico)} \\
 \phi = 0 &\quad \left\{ \begin{array}{l} \dot{\phi} < 0 \Rightarrow \gamma = 0 \Rightarrow \dot{\sigma} = E\dot{\epsilon} && \text{(descarga elastica)} \\ \dot{\phi} = 0 \Rightarrow \left\{ \begin{array}{l} \gamma = 0 \Rightarrow \dot{\sigma} = E\dot{\epsilon} && \text{(carga neutra)} \\ \gamma > 0 \Rightarrow \dot{\sigma} = E(\dot{\epsilon} - \dot{\epsilon}^p) && \text{(carga plastica)} \end{array} \right. \end{array} \right. && \text{(12)}
 \end{aligned}$$

El presente trabajo trata con los casos de carga neutra o plástica, por tanto, según las ecuaciones presentadas anteriormente se tiene que

$$\dot{\sigma} = H\dot{\epsilon}^p = \frac{EH}{E + H}\dot{\epsilon} = D\dot{\epsilon} \quad (13)$$

donde a D se le denomina el operador plástico tangente. Cabe aclarar que el operador $(\dot{\cdot})$ representa la derivada con respecto al tiempo de la variable analizada, en el caso de problemas que no dependen del tiempo, como el analizado en este documento, la variable t indica un pseudo-tiempo relacionado al proceso de carga cuasi-estático.

Establecido esto, el problema de localización de deformaciones se puede representar gráficamente como se muestra en la ilustración 5, donde el dominio correspondiente a la banda de localización se lo representa con la letra S .

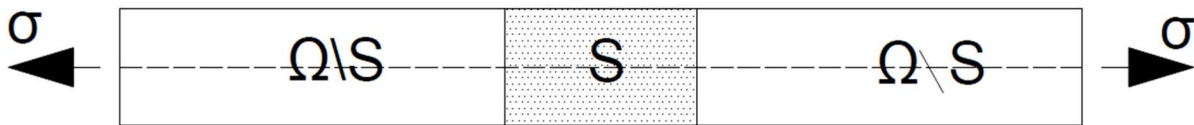


Ilustración 5.- Localización unidimensional

2.1.3. Enfoque variacional a la localización de deformaciones

En el momento de analizar problemas físicos se han formulado una cantidad diversa de métodos para su análisis partiendo desde distintos enfoques. Uno de estos enfoques se corresponde con el llamado principio variacional, formulado a lo largo del siglo XX, aunque sus orígenes se remontan hasta siglos anteriores. Un principio variacional es aquel que permite resolver un problema haciendo uso del cálculo de variaciones, a su vez se puede decir que esta rama del cálculo es aquella que se ocupa de la determinación de los extremos (máximos y mínimos) o de los valores estacionarios de las funciones (Rao, 2001).

En mecánica de sólidos el principio variacional, del cual parten los métodos, está relacionado estrechamente con el llamado principio de mínima acción. De forma general, este principio

establece que, dados dos puntos en el espacio, P1 y P2, y una partícula que se mueve desde P1 hasta P2, de la infinidad de recorridos posibles que dicha partícula puede realizar, el recorrido realmente (el que la naturaleza escoge) es aquel que minimiza la acción, es decir, donde la energía total (potencial más cinética) es mínima. Este procedimiento se conoce como procedimiento de Euler-Lagrange (Lanczos, 1962) y de este se deriva la ecuación

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{u}}(t, u, \dot{u}) = \frac{\partial L}{\partial u}(t, u, \dot{u}) \quad (14)$$

donde u representa a la posición de una partícula, \dot{u} representa la velocidad y $L(t, u, \dot{u})$ se conoce como lagrangiano de la partícula. El principio variacional puede ser extendido para el análisis de localización de deformaciones al analizar la minimización de la energía del sólido. Esta energía puede ser descompuesta en energía almacenada y energía disipada, según la ilustración 6.

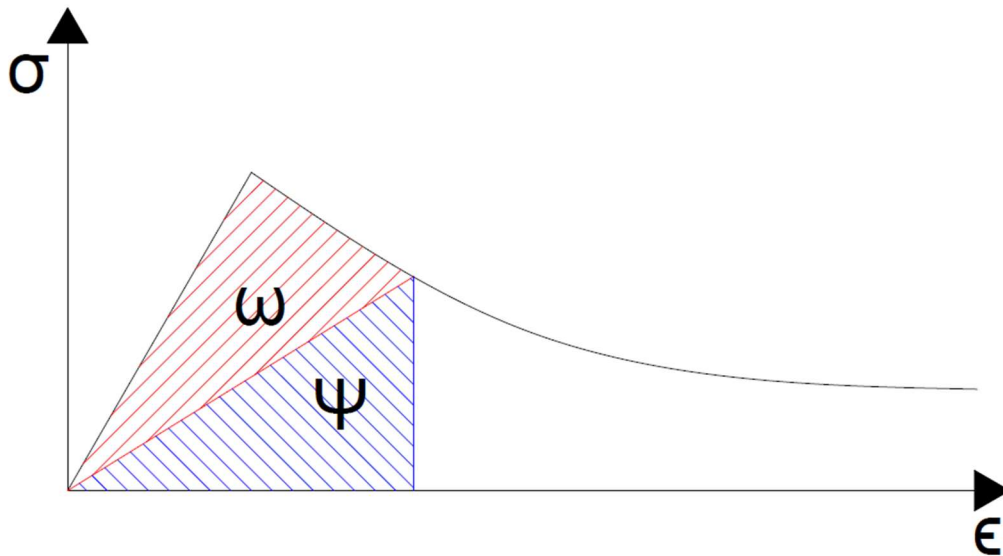


Ilustración 6.- Composición de la energía total de un sólido como el área bajo la curva esfuerzo-deformación. En rojo la energía disipada y en azul la energía almacenada

Definidos los términos de densidad de energía disipada ω y densidad de energía almacenada Ψ , el valor de la energía total del sólido unidimensional se determina como

$$W = \int_{\Omega} \omega(x) dx + \int_{\Omega} \Psi(x) dx . \quad (15)$$

El principio variacional para resolver el problema de localización de deformaciones se fundamenta por tanto en la minimización del potencial W .

2.1.4. Mecanismos de falla frágil

La falla frágil se produce cuando ninguna o casi ninguna plasticidad se presentan al momento de la fractura, esto ocurre cuando la tensión axial a la que está sometido un material supera la tensión de rotura σ_f , esto es:

$$\sigma > \sigma_f \quad (16)$$

Nótese que se utiliza σ_f para representar la tensión de rotura y diferenciarla de σ_y , la tensión de fluencia en el caso dúctil. La falla frágil puede ser modelizada haciendo uso de la conocida Mecánica de Fractura Elástico Lineal (LEFM) puesto que, en general, en estos casos se cumplen con las siguientes premisas:

- La relación tensión - deformación es elástica, o por lo menos casi lineal, hasta el momento de la fractura.
- El tamaño de la zona donde se produce el proceso de falla es despreciable en comparación con las demás dimensiones.

De esta manera el potencial de energía está compuesto únicamente para la energía almacenada (ilustración 7) y se reduce a

$$W = \int_0^L \Psi^e(\epsilon^e) dx = \int_0^L \frac{1}{2} \epsilon^e \sigma dx = \int_0^L \frac{1}{2} E(\epsilon^e)^2 dx. \quad (17)$$

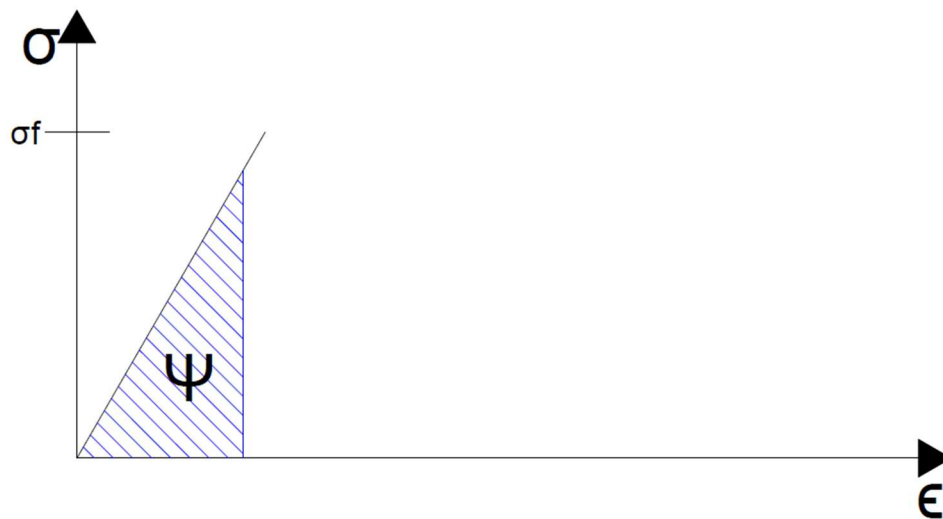


Ilustración 7.- Potencial de energía almacenada en el caso de fractura frágil

Este tipo de falla es característico en materiales tales como el cristal y materiales cerámicos, además de caracterizar de forma bastante aproximada el comportamiento del hormigón, que,

de hecho, es un material cuasi-frágil, por lo que su estudio aún es de especial interés para la ingeniería civil.

2.1.5. Mecanismo de falla dúctil

De forma contraria al caso de fallo frágil, en donde no se presentan deformaciones plásticas, en el fallo dúctil es condición necesaria la presencia de dichas deformaciones. La fractura dúctil se produce por una degradación progresiva de la rigidez y resistencia del material, tal como se aprecia en el diagrama bilineal de la ilustración 8, que deviene en la deformación plástica. La fractura dúctil es un proceso complejo y su descripción requiere una explicación adecuada de la nucleación y el movimiento de las dislocaciones en las proximidades de la punta de la grieta (Ganchenkova & Nieminen, 2010).

El caso de falla dúctil ha impulsado una serie de investigaciones y su estudio de cierta forma ha servido de motor que promueve el avance dentro del campo de la mecánica de fractura. Ha habido aproximaciones a este problema desde la perspectiva clásica, como la que se mencionó en secciones anteriores. Sin embargo, para los motivos de esta investigación, se dará énfasis a la aproximación variacional del problema, que ya ha sido presentada.

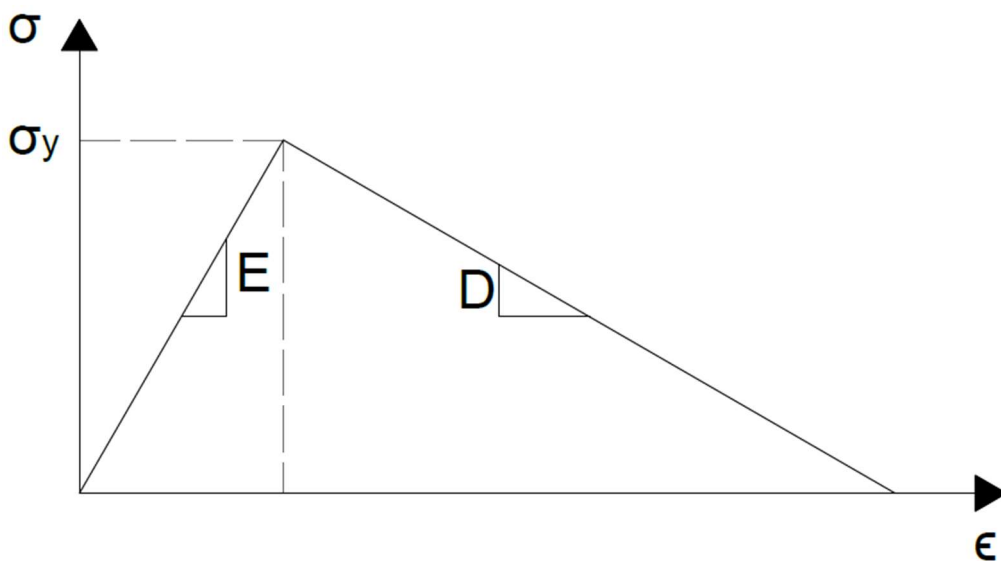


Ilustración 8.- Curva esfuerzo-deformación del modelo constitutivo con ablandamiento

Empleando este enfoque, resolver el problema unidimensional es equivalente a resolver el problema de minimización de la energía, que para el caso de falla dúctil es el potencial formado por la descomposición de la energía de la ilustración 9.

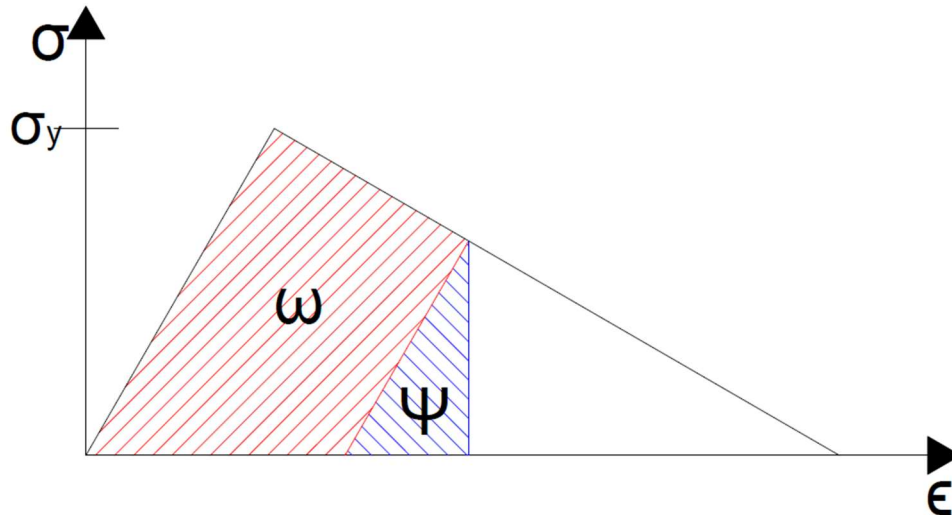


Ilustración 9.- Descomposición de la energía en almacenada (azul) y disipada (rojo) para el modelo con ablandamiento y sin pérdida de rigidez

La energía del sólido es, por tanto,

$$\begin{aligned}
 W &= \int_{\Omega} \Psi(x)dx + \int_{\Omega} \omega(x)dx \\
 &= \int_0^L \frac{1}{2} E(\epsilon^e)^2 dx + \int_{\frac{L-h}{2}}^{\frac{L+h}{2}} \frac{1}{2} H(\epsilon^p)^2 dx + \int_{\frac{L-h}{2}}^{\frac{L+h}{2}} \sigma_y(\epsilon^p) dx \quad (18)
 \end{aligned}$$

En esta ecuación los dos últimos términos están relacionados directamente con la disipación de energía y, como se puede observar, la integral incluye solo la zona del dominio comprendida por la banda de localización, que es la zona donde efectivamente se produce la localización de deformaciones. Alternativamente, se puede calcular la energía almacenada y disipada a partir de las áreas del trapecio y triángulo bajo la curva esfuerzo-deformación (ilustración 9) lo cual conduce a la ecuación

$$\begin{aligned}
 W &= \int_{\Omega} \Psi(x)dx + \int_{\Omega} \omega(x)dx \\
 &= \int_0^L \frac{1}{2} E(\epsilon^e)^2 dx + \int_{\frac{L-h}{2}}^{\frac{L+h}{2}} \frac{1}{2} \left(\sqrt{\epsilon_L^2 + \sigma_y^2} + \sqrt{(\epsilon^e)^2 + \sigma^2} \right) (\epsilon^p) * \sin(\theta) dx \quad (19)
 \end{aligned}$$

En relación con la localización, el problema de fallo dúctil se puede estudiar desde las dos perspectivas analizadas en capítulos anteriores, esto es discontinuidad fuerte y discontinuidad débil. Aún más, ha sido demostrado (Simo et al., 1993) que bajo ciertas condiciones se puede considerar que el modelo teórico discreto (discontinuidad fuerte) puede

considerarse como el caso límite del modelo teórico continuo (discontinuidad débil) cuando el ancho de la banda es nulo.

2.2. Métodos numéricos

En muchas ocasiones las ecuaciones que modelan el comportamiento de un sistema físico o describen algún fenómeno al que están sujetos ciertos materiales, no pueden ser resueltas de forma analítica, por lo que encontrar una solución exacta de la forma tradicional no es viable. Es en estos casos donde la ayuda de métodos numéricos se vuelve necesaria. Un método numérico es un método computacional aproximado para resolver un problema matemático que a menudo no tiene solución analítica (Hahn & Valentine, 2013).

2.2.1. Métodos numéricos en la ingeniería civil

Al igual que en otras áreas de la ciencia y la ingeniería, la ingeniería hace uso continuo de los métodos numéricos, en especial gracias al avance de las ciencias de la computación y a la mejora de las capacidades de procesamiento. Existe una gran variedad de métodos existentes, como por ejemplo el método de elementos finitos (FEM), que han sido probados con éxito.

Para el presente trabajo se hará mención a los llamados métodos *physics-based*, estas se enmarcan dentro de un área aún mayor que es el Machine Learning (ML), que será presentado más adelante. Las simulaciones numéricas *physics-based* se han hecho indispensables en aplicaciones de ingeniería civil, como la mitigación de riesgos sísmicos, el diseño y análisis de estructuras y el monitoreo estructural.

Los ingenieros civiles y los científicos pueden ahora utilizar modelos sofisticados para aplicaciones reales, con simulaciones ultrarrealistas con millones de grados de libertad, gracias a los avances de los ordenadores de alto rendimiento. Sin embargo, en el sector de la ingeniería civil estas simulaciones requieren demasiado tiempo para incorporarlas plenamente a un proceso de diseño iterativo. A menudo se limitan a las fases finales de validación y certificación, mientras que la mayoría de los procesos de diseño se basan en modelos más sencillos. Acelerar las simulaciones complejas es un problema importante a tener en cuenta, ya que facilitaría la aplicación de herramientas numéricas a lo largo de todo el proceso de diseño.

El desarrollo de métodos numéricos para simulaciones rápidas también permitiría nuevas aplicaciones de los modelos, como la mejora de la productividad de la construcción, que aún no se ha utilizado plenamente debido a la complejidad de los modelos (Vadyala et al., 2022).

La cuantificación de la incertidumbre es otro ejemplo crítico de análisis que podría ser factible si los costes de simulación se redujeran sustancialmente. En efecto, el entorno físico del sistema que generalmente se desconoce, afecta a los valores de interés monitoreados en simulaciones numéricas. En algunas situaciones, estas incertidumbres impactan significativamente los resultados de las simulaciones, por lo que es necesario las distribuciones de probabilidad de las cantidades de interés para garantizar la fiabilidad del resultado. Ni un método basado únicamente en ML ni un método basado únicamente en el conocimiento científico puede considerarse suficiente para aplicaciones científicas y técnicas complejas.

2.2.2. Machine Learning

Machine Learning (ML) se refiere a la capacidad de un sistema para adquirir e integrar conocimientos a través de observaciones a gran escala, y para mejorar y ampliarse a sí mismo mediante el aprendizaje de nuevos conocimientos en lugar de ser programado con esos conocimientos (Woolf, 2009).

Las Redes Neuronales (NNs) son modelos de ML para expresar la relación entrada-salida de una forma

$$Y = Y^{NN} = W^T \phi_h(B^T \bar{x}) + \eta \quad (20)$$

donde $\bar{x} = [x; 1]$, Y es la variable objetivo (de salida), mientras que x es la variable de entrada, Y^{NN} es la variable de salida predicha por la NN. La función de activación de la variable de entrada es ϕ_h , la matriz de pesos de transición es B , la matriz de pesos de salida es W , y η es un error desconocido debido a errores de medición o de modelado. Dentro de las matrices de pesos, los términos de sesgo se definen complementando la variable de entrada x con un valor unitario en las presentes notaciones. En la anterior la variable objetivo es una combinación lineal de determinadas funciones básicas parametrizadas por B . Un diseño de red neuronal con profundidad K capas se define en la ecuación

$$Y \approx Y^{NN} = W^T \phi_{k-1}(\phi_{k-2}(\dots \phi_1(B_1^T \bar{x}))) \quad (21)$$

donde ϕ_k y B_k son la función no lineal elemento a elemento (función de activación) y la matriz de pesos para la capa K-ésima y W es la matriz de pesos de salida. La ilustración 10 muestra un esquema de red neuronal con una capa oculta.

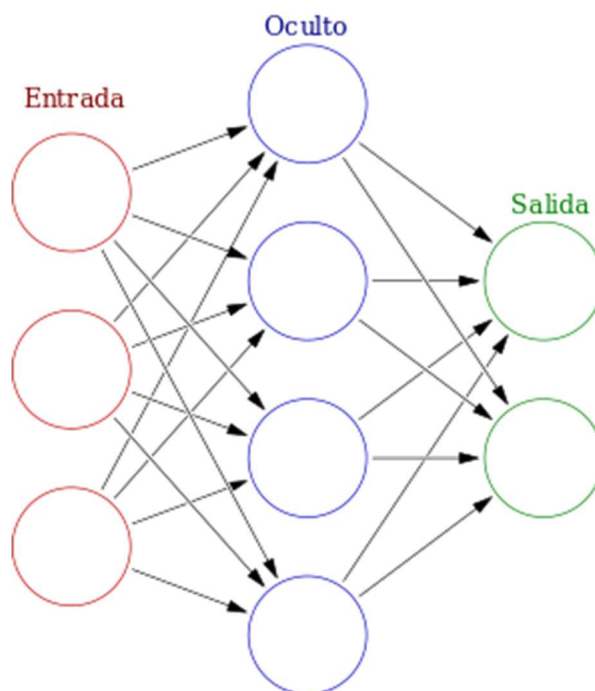


Ilustración 10.- Esquema de red neuronal (*Colored Neural Network*, 2019)

2.2.3. Physics Informed Neural Networks (PINN)

El método Physics-based ML puede combinar los conocimientos que ya tenemos, como los algoritmos de avance y optimización basados en la física. La mecánica de entrenamiento de una red basada en la física es similar a la de cualquier red neuronal; se basa en un conjunto de datos, un optimizador y una diferenciación automática (Griewank & Walther, 2008) para calcular los gradientes.

La modelización de procesos físicos descritos por EDP ha mejorado gracias a las PINN de forma significativa, estas fueron propuestas en investigaciones recientes (Raissi et al., 2018). Se basa en técnicas que incluyen el uso de redes neuronales artificiales y métodos de ajuste de ecuaciones diferenciales, que gobiernan el sistema físico modelado, usando varios puntos del dominio que se denominan puntos de colocación (Katsikis et al., 2022). El comportamiento de sistemas físicos complicados se aprende dentro de la red neuronal minimizando el residuo de las PDE subyacentes a través de la optimización de factores de la red.

Considerando que el problema analizado en este trabajo se gobierna por la ecuación de energía y las condiciones de contorno:

$$\min\{W | u = u^* \text{ en } \Gamma_u\} \quad (22)$$

Esta ecuación de energía del sólido se la puede aproximar como una sumatoria al multiplicar la densidad de energía por un peso en los puntos de integración (los puntos de colocación), de esta manera:

$$W(u) \approx \mathcal{L}(p) = \sum (\omega(x_i) + \Psi(x_i))w_i \quad (23)$$

Donde p representa los parámetros entrenables de la red. Discretizada de esta manera se puede usar el método PINN para minimizar la energía a través del entrenamiento de los parámetros de la red (pesos y sesgos). Considerando condiciones de contorno tipo Dirichlet la función de la pérdida se define como

$$LOSS = W(\epsilon) + (\hat{u}(x_0) - u_0) + (\hat{u}(x_L) - u_L) \quad (24)$$

El entrenamiento se realiza siguiendo el esquema indicado en la ilustración 11.

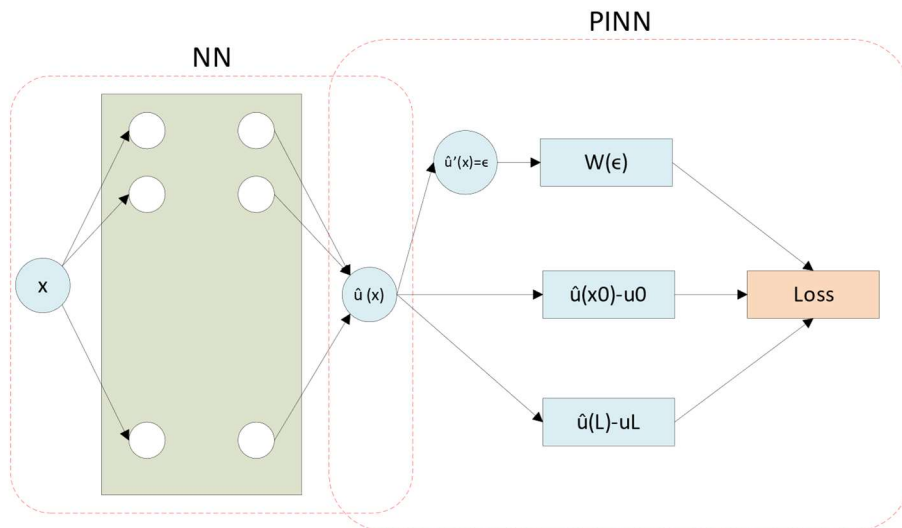


Ilustración 11.- Arquitectura de la NN regular (izquierda) en combinación con PINN (derecha).

3. Materiales y Metodología

3.1. Materiales

Por la naturaleza del presente trabajo los materiales utilizados no corresponden a datos reales provenientes de ensayos de laboratorio, sino que se corresponden con una serie de modelos

teóricos previamente desarrollados para el estudio de los modos de falla dúctiles y que se recogen en el presente trabajo.

En lo referente al primer objetivo, la modelización del mecanismo de falla dúctil, se tiene como principal modelo teórico la propuesta variacional de Alessi (Alessi, 2013) posteriormente ampliada por Ulloa et. al (Rodriguez et al., 2018; Ulloa et al., 2016). Los detalles del enfoque variacional ya han sido presentados en secciones anteriores.

Partiendo de este enfoque variacional se hace uso de las propuestas en lo referente a PINN de los autores (Raissi et al., 2018) y (Samaniego et al., 2020) que han demostrado la posibilidad de usar un enfoque de minimización de la energía, esto es partiendo de enfoques variacionales, para resolver EDPs.

La PINN es programada en el lenguaje de programación Python. Además, se utiliza la librería TensorFlow 2.11.0, una biblioteca gratuita y de código abierto para el aprendizaje automático y la inteligencia artificial. Puede utilizarse en una amplia gama de tareas, pero se centra especialmente en el entrenamiento y la inferencia de redes neuronales profundas (Abadi et al., 2016).

3.2. Metodología

Se utilizo un enfoque variacional a la modelización de falla dúctil, es decir a la minimización de la energía. En el marco de esta formulación, se plantea el mismo problema descrito por Rodriguez y Ulloa (2015) de describir la evolución una barra de longitud L , sujeta a un proceso de carga cuasi estático en el que se controlan desplazamientos. El esquema de la barra y las condiciones de carga a las que es sometida se pueden observar en la ilustración 12. Esta prueba a tensión para un material dúctil resalta los principales conceptos de modelización de la fractura dúctil de materiales (Alessi, 2013). Por simplicidad, se asume un material sin deformaciones plásticas ni daño al inicio de las simulaciones.

La metodología general utilizada para el desarrollo de este trabajo se detalla a continuación:

- Definición de los componentes y principios sobre los que se desarrolla cada formulación.
- Descripción de los modelos matemáticos.
- Implementación de algoritmos numéricos basados en la aplicación de PINN en Python mediante TensorFlow.

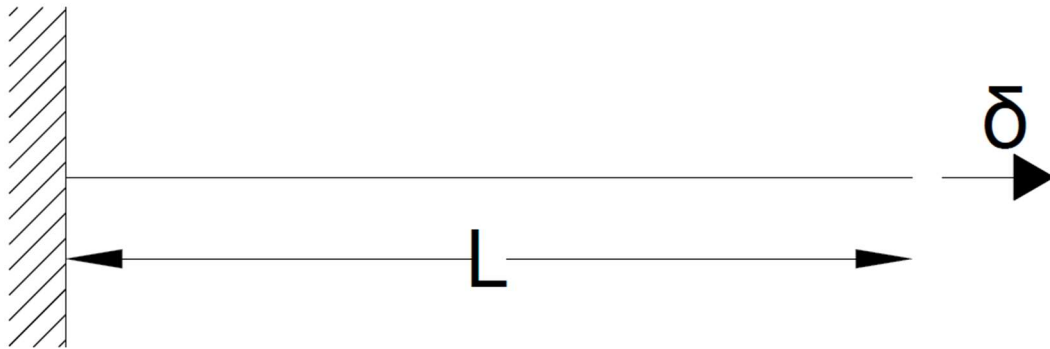


Ilustración 12.- Definición del problema unidimensional

4. Configuración geométrica y mecánica del caso de estudio

4.1. Planteamiento del caso de estudio

Como se ha mencionado, el presente trabajo lidiará con el problema unidimensional cuasi-estático, este problema a pesar de su sencillez permite experimentar y poner a prueba las bases teóricas sobre las que se fundamenta la mecánica de fractura y la localización de deformaciones. En este caso la sencillez constituye una ventaja debido a que permite al investigador comprender de mejor manera la forma en cómo funciona tanto el método planteado, es decir la PINN, así como profundizar de mejor manera en los conceptos subyacentes.

Las ecuaciones de gobierno que controlan en general el comportamiento mecánico ya han sido introducidas en secciones anteriores, a fin de plantear claramente el problema a analizar se presentan nuevamente y de forma unidimensional:

$$\frac{d\sigma}{dx} = 0 \quad \text{Ecuación de balance} \quad (25)$$

$$\frac{du}{dx} = \epsilon \quad \text{Ecuación de equilibrio} \quad (26)$$

$$u = u^* \text{ en } \Gamma_u \quad \text{Condiciones de contorno} \quad (27)$$

A partir de estas ecuaciones, y en conjunto con las relaciones constitutivas (que son propiedades del material), se pueden analizar los distintos casos de estudio en su formulación fuerte. Como se analizó, el método variacional resuelve el problema a través de un proceso

de minimización de la energía del sólido, este potencial de energía ya ha sido presentado en la sección 2.1 para los casos frágil y dúctil.

4.2. Configuración geométrica y propiedades mecánicas de la barra 1D

Considerando el problema de la ilustración 12, el dominio unidimensional es el eje de la barra, que se representará por la letra Ω , por tanto Ω está constituido por todos los puntos entre 0 y L ($\Omega = [0, L]$). Esta barra posee una sección transversal constante A y no está sometida a fuerzas volumétricas, además la barra está sometida a un desplazamiento δ en el extremo.

Las propiedades mecánicas de la barra se detallan en la siguiente tabla:

PROPIEDAD MECÁNICA	VALOR
Módulo de elasticidad (E)	2
Tensión de fluencia (σ_y) y Tensión de rotura (σ_f)	1
Operador elastoplástico tangente (D)	-0.2

4.3. Mecánica de la barra 1D, caso frágil

En el caso frágil, como ya se ha hecho mención en capítulos anteriores, la barra se caracteriza por no presentar fluencia plástica, esto provoca un comportamiento caracterizado por un aumento en la tensión interna del material que se corta abruptamente cuando esta falla, es decir, pierde la capacidad para resistir más carga. La relación constitutiva de este proceso es

$$\sigma = E\epsilon \quad (28)$$

que para el caso de estudio se reduce a

$$\sigma = 2\epsilon$$

Considerando el enfoque variacional, que será utilizado en este trabajo, ya ha sido introducida la función de densidad de energía en el caso frágil, esta es:

$$W = \int_0^L \Psi^e(\epsilon^e) dx = \int_0^L \frac{1}{2} \epsilon^e \sigma dx = \int_0^L \frac{1}{2} E (\epsilon^e)^2 dx \quad (17)$$

Por tanto, la solución al problema se la obtiene a través de:

$$\min\{W \mid u = u^* \text{ en } \Gamma_u\} \quad (29)$$

La anterior función de densidad de energía se corresponde directamente a la energía potencial elástica, puesto que en el proceso ideal de falla frágil no existen procesos a través de los cuales se disipe energía durante la carga (tales como plasticidad, daño, calor, etc.), sino que el material disipa la totalidad de su energía almacenada de manera abrupta al producirse el fallo.

4.4. Mecánica de la barra 1D, caso dúctil

En este caso, donde la presencia de deformación plástica es una característica fundamental, se pueden presentar tres casos que ya han sido mencionados:

- Endurecimiento: La tensión aumenta lentamente con grandes aumentos en la deformación.
- Plasticidad perfecta: La tensión permanece constante con los aumentos de deformación.
- Ablandamiento: La tensión disminuye al presentarse aumentos en la deformación.

Para el presente trabajo se considera el último caso, el de ablandamiento. En un proceso de fallo localizado en realidad se observan dos situaciones de carga: una descarga elástica fuera de la banda de localización y una carga plástica dentro de esta. Este comportamiento implica la necesidad de plantear dos relaciones constitutivas que, dependiendo del umbral de la ubicación de un punto material (fuera o dentro de la banda), actuará en favor de alguno de los dos casos de carga. Estas relaciones son:

$$\sigma = E\epsilon \text{ en } \Omega \setminus S \text{ (Fuera de la banda } S) \quad (30)$$

$$\sigma = D(\epsilon - \epsilon_L) + \sigma_y \text{ en } S \text{ (Dentro de la banda } S) \quad (31)$$

Reemplazando los valores de las propiedades del material en las relaciones anteriores se llega a:

$$\sigma = 2\epsilon$$

$$\sigma = -0.2(\epsilon - 0.5) + 1$$

De igual manera la función de la densidad de energía para este caso de estudio ya ha sido introducida en secciones anteriores y es:

$$A = \int_0^L \Psi^e(\epsilon^e) + \Psi^p(\epsilon^p) dx = \int_0^L \frac{1}{2} \epsilon^e \sigma + \frac{1}{2} H \epsilon^p dx$$

$$= \int_0^L \frac{1}{2} E (\epsilon^e)^2 + \frac{1}{2} H (\epsilon - \epsilon^e) dx \quad (32)$$

$$D = \int_0^L \varphi^p(\epsilon^p) = \int_0^L \epsilon^p \sigma_y \quad (33)$$

$$W = A + D \quad (34)$$

La función W es la que corresponde a ser minimizada, por tanto, el problema a resolver es:

$$\min\{W | u = u^* \text{ en } \Gamma_u\} \quad (29)$$

En la función de densidad de energía se puede observar la presencia de una densidad de energía plástica, así como una función de disipación, ambos términos son necesarios debido a la presencia de deformaciones plásticas.

5. Modelización de la barra 1D

5.1. Solución analítica del problema unidimensional

Por la naturaleza del problema, es posible encontrar directamente la solución analítica del mismo a partir de las ecuaciones de gobierno unidimensionales 25, 26, y 27, estas ecuaciones se pueden alterar para obtener las siguientes igualdades:

$$E \epsilon^e = D(\epsilon - \epsilon_L) + \sigma_y \quad (35)$$

$$(L - h) * \epsilon^e + h * (\epsilon - \epsilon^e) = \delta \quad (36)$$

$$\epsilon^e = \frac{u_1}{l_2} = \frac{u_2}{l_2} \quad (37)$$

$$\epsilon - \epsilon^e = \epsilon^p = \frac{[[u]]}{h} \quad (38)$$

En este sistema las cantidades desconocidas son u_1, u_2 y $[[u]]$, esta última se corresponde con el salto en el campo de desplazamiento producto de la localización de deformaciones, la

geometría de los desplazamientos como de la barra deformada se presenta en el esquema de la ilustración 13.

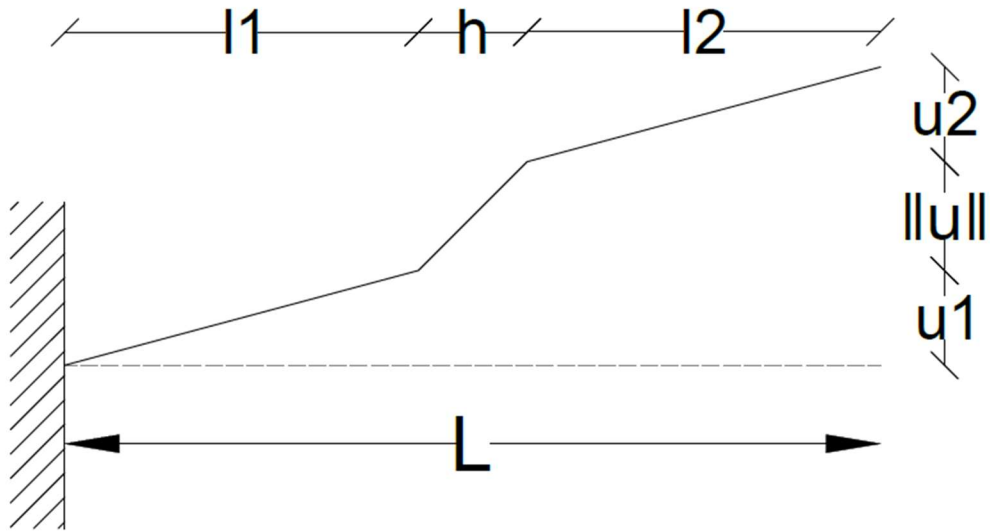


Ilustración 13.- Esquema del desplazamiento en la barra producto de la localización

Resolviendo el sistema se obtienen los siguientes resultados de desplazamiento, deformación y tensión:

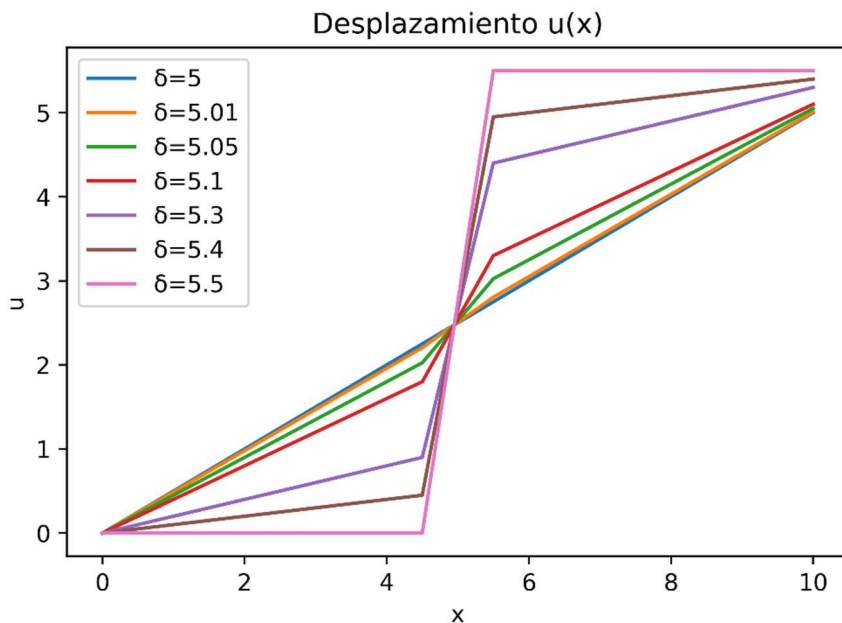


Ilustración 14.- Desplazamiento de la barra, solución analítica

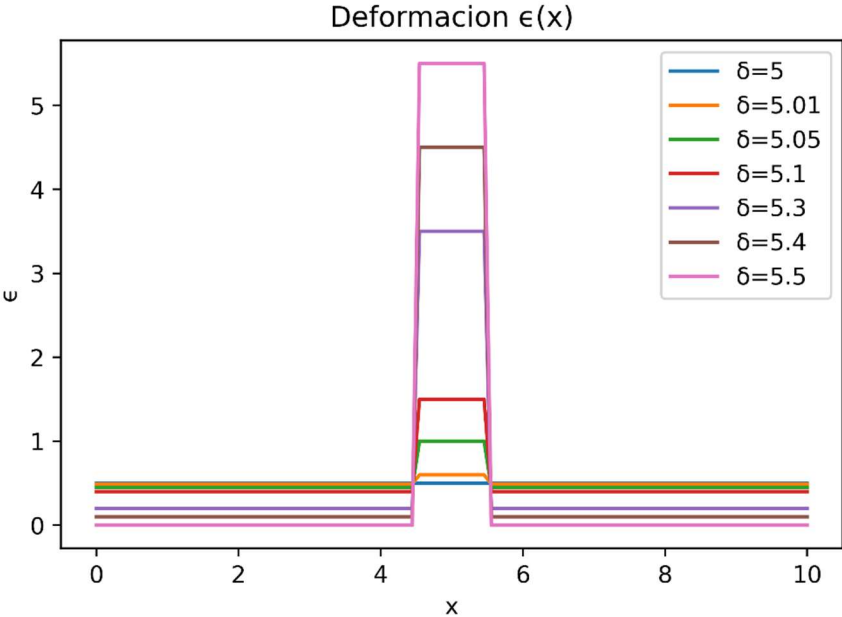


Ilustración 15.- Deformación de la barra, solución analítica

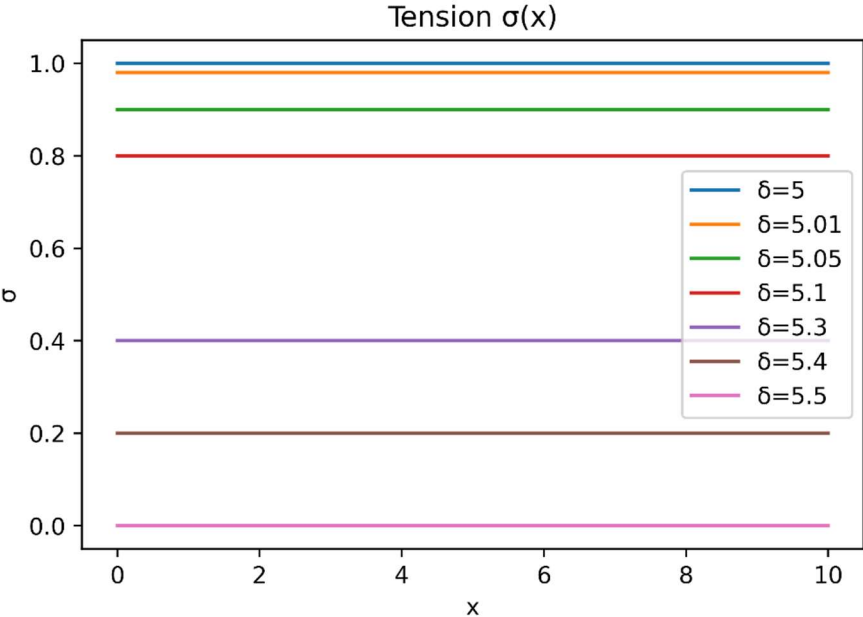


Ilustración 16.- Tensión en la barra, solución analítica

Las ilustraciones 14, 15 y 16 representan la evolución de la deformación plástica en la zona correspondiente a la banda de localización, se puede observar que para el último estado de carga toda la deformación de la barra se concentra en la banda, es decir, corresponde a

deformación y plástica, por tanto, este estado se corresponde con la rotura del material. El código para la solución analítica se encuentra completo en el Anexo A.

5.2. Implementación de la PINN

La implementación de la PINN se inspiró en varias arquitecturas de redes neuronales presentes en la literatura, en particular se puede mencionar el modelo presentado por Katsikis et. al (2022). Dicho modelo presenta la ventaja de dividir de manera clara las clases que se usan en la implementación, lo que facilita el entendimiento del comportamiento de la PINN.

De igual manera, recientemente se han hecho varias investigaciones orientadas a la forma de implementar redes neuronales en la resolución de problemas de mecánica de sólidos (Goswami et al., 2020; Haghghat et al., 2021). Sin embargo, a pesar de mostrar implementaciones novedosas a la par que rigurosas, estos hacen uso de modelos más complejos como phase-fields, o se enfocan en problemas de obtención de parámetro físicos (modelos subrogados e inversos).

El modelo implementado está basado en tres módulos que serán detallados a continuación. La primera clase es GradientLayer(), forma parte del módulo layer y es la encargada de realizar el proceso de diferenciación automática, este es de esencial importancia dentro del contexto de redes neuronales puesto que permite realizar la optimización por medio de descenso de gradiente (como el método SGD) además de ser útil a la hora de plantear las EDP o EDO del problema a analizar ya que implican necesariamente la representación por derivadas de ciertas variables.

A continuación, se muestran algunas líneas de código correspondientes a este módulo:

```
class GradientLayer(tf.keras.layers.Layer):
    def __init__(self, model, **kwargs):
        self.model = model
        super().__init__(**kwargs)
    def call(self, x, E, D, eps_l, sigma_y, L, h):
        with tf.GradientTape() as g:
            g.watch(x)
            with tf.GradientTape() as gg:
                gg.watch(x)
```

```

        u = self.model(x)
        eps = gg.gradient(u, x)
        sigma=E*eps
        ds_dx=g.gradient(sigma, x)
        return u, eps, sigma, ds_dx

```

En las líneas se puede observar que es dentro de este módulo donde se realiza el cálculo de la deformación como el gradiente del desplazamiento, a su vez se realiza la definición de las ecuaciones constitutivas dentro del módulo, es decir la función sigma, que dependerá del caso, ya sea plástico o elástico, que se está tratando.

Definido el módulo de diferenciación automática se puede presentar el módulo central de la PINN, este es el módulo network, la clase que se define este módulo es PINN(). A continuación, se presentan líneas de código de este módulo:

```

class PINN:
    def build(self, num_inputs=1, layers=[10,10,10], activation='tanh',
num_outputs=1):
        inputs = tf.keras.layers.Input(shape=(num_inputs,))
        x = inputs
        for layer in layers:
            x = tf.keras.layers.Dense(layer, activation=activation,
kernel_initializer='he_normal')(x)
        out=tf.keras.layers.Dense(num_outputs,kernel_initializer='he_normal')
(x)
        u_model = tf.keras.models.Model(inputs=inputs, outputs=out)
        grads = GradientLayer(u_model)
        x_1 = tf.keras.layers.Input(shape=(1,))
        x_2 = tf.keras.layers.Input(shape=(1,))
        x_3 = tf.keras.layers.Input(shape=(1,))
        u, eps, sigma, ds_dx = grads(x_1, E=self.E, D = self.D,
eps_l=self.eps_l, sigma_y=self.sigma_y, L=self.L, h=self.h)

```

```

        u_1 = (0.5*(sigma**2/self.E)+0.5*self.H*(eps-sigma/self.E)**2) +
self.sigma_y*(eps-sigma/self.E)

        u_2 = u_model(x_2)

        u_3 = (u_model(x_3) - self.delta)

        return u_model, tf.keras.models.Model(

            inputs=[x_1, x_2, x_3] , outputs=[u_1, u_2, u_3])

```

La clase PINN es el núcleo central donde se desarrolla el método y la modelización, se comienza la clase con la declaración de las variables y parámetros físicos que ya han sido presentados anteriormente en el documento, estos parámetros también variarán dependiendo del caso, plástico o elástico, que se esté estudiando. Una vez declaradas se procede con la función `build()`, esta función construye la red neuronal en sí, como se observa, se crean en realidad dos redes neuronales, el primero es el modelo `u_model()` que tiene un input `x`, que son las coordenadas de la barra unidimensional, y un output `u` que corresponden a los desplazamientos.

Este modelo a su vez está conectado con el segundo modelo, la red `PINN()`, que es la que define al final de la clase, este modelo recibe tres inputs, las coordenadas `x` (`x1`), un vector cuyos elementos son la coordenada cero (`x2`), y vector con elementos de coordenada `L` (`x3`). Se calculan los gradientes haciendo uso de la clase `GradientLayer()`, con los resultados de esta se calcula la densidad de energía (`u1`) y las condiciones de contorno (`u2` y `u3`), estos vectores sirven de output para el modelo. Lo destacable en este punto es que entrenar el modelo `PINN()` implica necesariamente entrenar el modelo `u_model()` lo que garantiza que este último cumpla con la condición de minimización y las condiciones de contorno.

El tercer y último módulo que se define es el módulo `main`, en este se definen las variables y se procede al cálculo, este módulo presenta varias líneas de código, a continuación, se presentan las más relevantes:

```

x_1 = np.linspace(0,L,num_train_samples).reshape((num_train_samples,1))
x_2 = np.zeros((num_train_samples, 1))
x_3 = L*np.ones((num_train_samples, 1))

u_1 = np.zeros((num_train_samples, 1))           # u_1 = 0
u_2 = np.zeros((num_train_samples, 1))           # u_2 = 0
u_3 = np.zeros((num_train_samples, 1))           # u_3 = 0

```

```

x_train = [x_1, x_2, x_3]

y_train_target = [u_1, u_2, u_3]

u_model, PINN_model = PINN(E, D, eps_1, sigma_y, L, d, h, cond).build()

PINN_model.compile(optimizer='adam',

                    loss=tf.keras.losses.mse,

                    metrics=[tf.keras.metrics.mse],

                    )

model_history=PINN_model.fit(x_train,y_train_target,batch_size=50000,
epochs=5000, callbacks=[callbacks])

x_test = np.linspace(0, L, num_test_samples)

u = u_model.predict(x_test, batch_size=num_train_samples)

```

Primero se definen los inputs y outputs, se generan los modelos y se compila el modelo `PINN()`, en esta se elige el método de optimización que será implementado. En este trabajo se utiliza el método Adam, este es un algoritmo para la optimización basada en gradientes de primer orden de funciones objetivo estocásticas, basado en estimaciones adaptativas de momentos de orden inferior (Kingma & Ba, 2014). Se ha elegido este método por ser utilizado en la literatura reciente con buenos resultados (Katsikis et al., 2022).

Por último, se entrena la red con el método `.fit()`, aquí se elige tanto el número de épocas como el tamaño de muestreo, que en este caso es la totalidad de la muestra, además se puede seleccionar algún callback que, en el contexto de la librería TensorFlow actúa como una tolerancia para detener el entrenamiento cuando se ha alcanzado un mínimo. Por último, una vez se ha entrenado el modelo, se obtiene la respuesta con el método `.predict()`.

5.3. Modelización de la barra 1D, caso frágil

Considerando el planteamiento matemático que se ha presentado la barra se puede modelizar utilizando la siguiente clase `GradientLayer()`:

```

class GradientLayer(tf.keras.layers.Layer):

    def __init__(self, model, **kwargs)

        self.model = model

        super().__init__(**kwargs)

```



```

def call(self, x, E):
    with tf.GradientTape() as g:
        g.watch(x)
        with tf.GradientTape() as gg:
            gg.watch(x)
            u = self.model(x)
            eps = gg.gradient(u, x)
            sigma = E*eps
        ds_dx =g.gradient(sigma, x)
    return u, eps, sigma, ds_dx

```

Como se observa, la relación constitutiva esta únicamente definida por la ecuación 28. El módulo network presenta a su vez las siguientes líneas de código:

```

u, eps, sigma, ds_dx = grads(x_1, E=self.E)

# POTENCIAL DE ENERGIA
u_1 = 0.5*self.E*eps**2

# CONDICIÓN INICIAL OUTPUT
u_2 = 10*u_model(x_2)

# CONDICIÓN DE CONTORNO OUTPUT
u_3 = 10*(u_model(x_3) - self.delta)

```

En estas se calcula la función de densidad de energía de la ecuación 17, que será minimizada. El código completo se presenta en el Anexo B. La red se implementa con las siguientes características:

- Número de puntos de muestra (Collocation Points): 1000
- Numero de épocas: 10000 para el primer desplazamiento y 1000 en los siguientes.
- Callbacks: No se implementan callbacks
- Capas ocultas de la red neuronal: 0 capas ocultas, solo se considera la capa input y la capa output.
- Función de activación: Sin función de activación

Los resultados obtenidos se presentan en las ilustraciones 17-19, se han analizado los casos de carga con desplazamiento sucesivos de 0.25, 0.5, 1, 2, 3, 4 y 5 en el extremo.

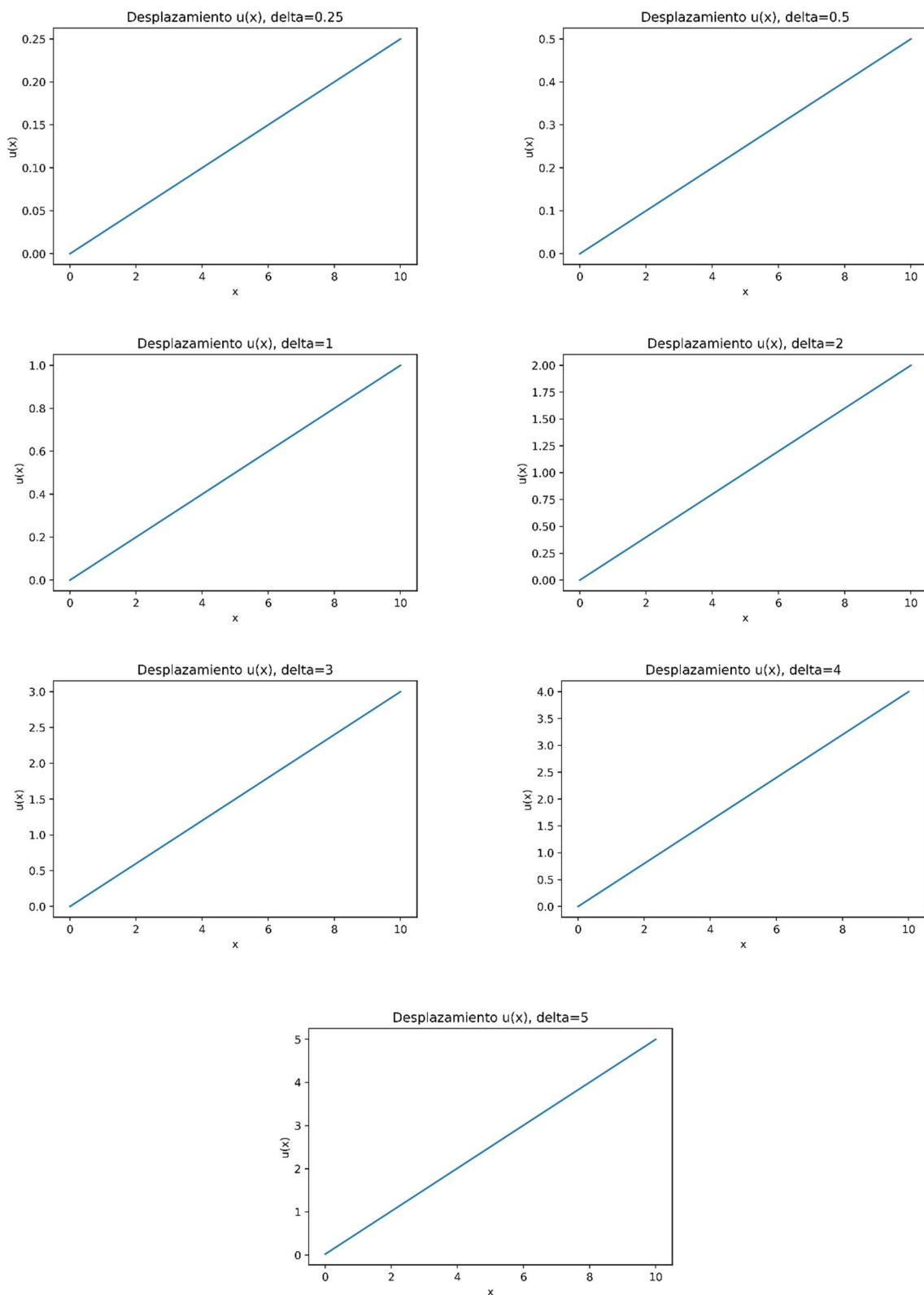


Ilustración 17.- Desplazamiento para distintos casos de desplazamiento en el extremo

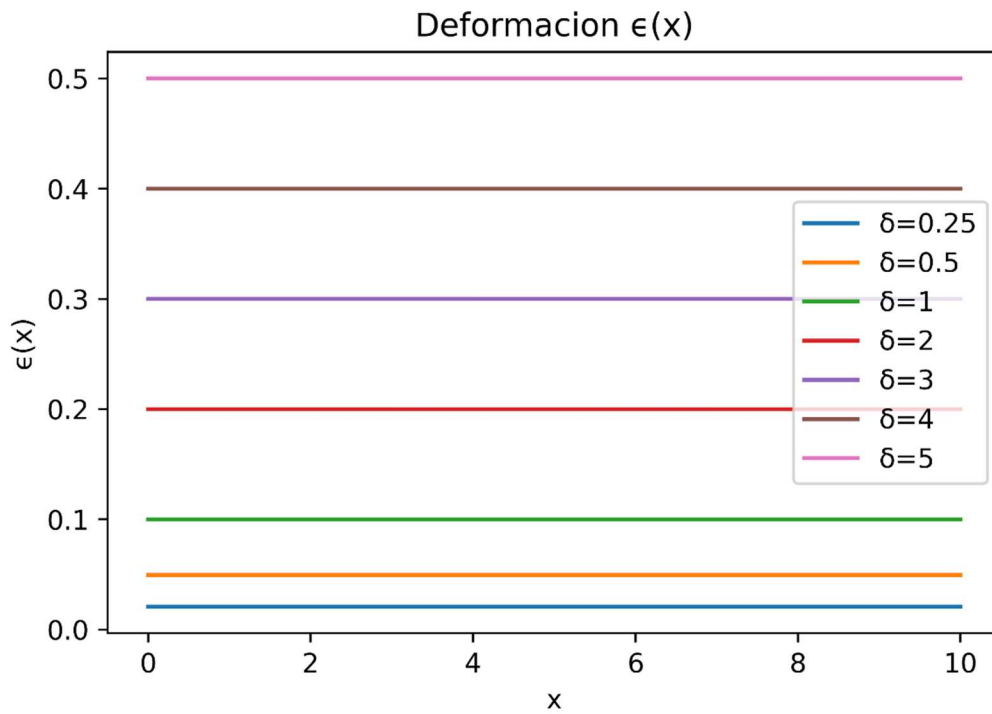


Ilustración 18.-Deformación para distintos casos de desplazamiento en el extremo

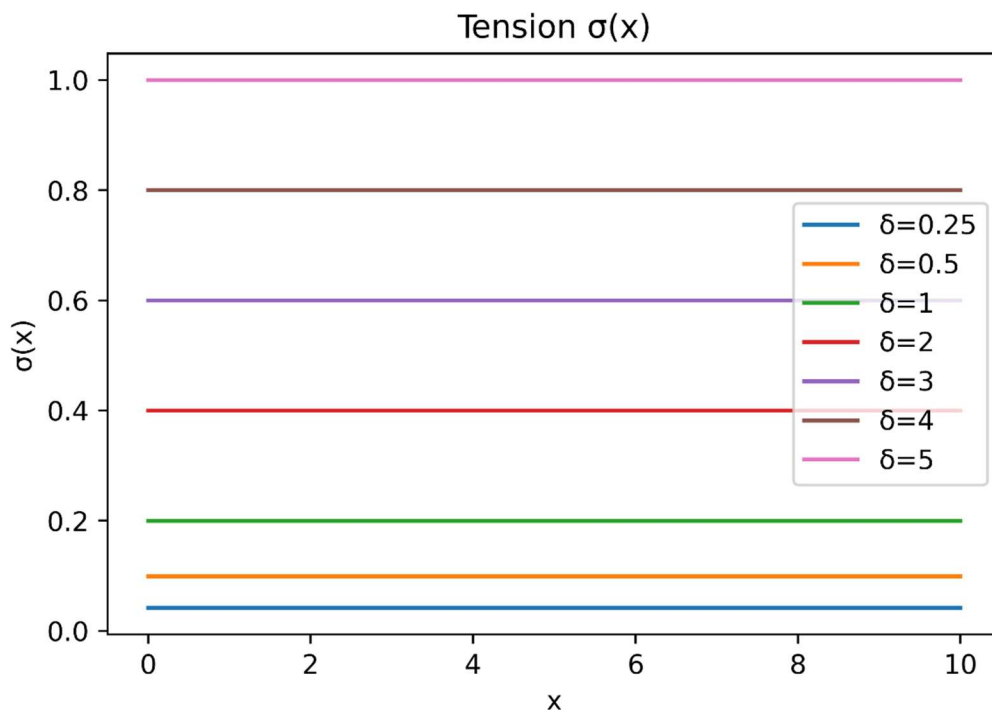


Ilustración 19.- Tensión para distintos casos de desplazamiento inicial en el extremo

Una vez alcanzada la tensión máxima se producirá la fractura donde se disipará la energía almacenada de acuerdo con la teoría de Griffith. En las ilustraciones anteriores se observa como la implementación realizada captura el comportamiento elástico del material, es decir, la rama lineal antes del fallo, cabe señalar que esta aproximación puede ser utilizada para directamente simular la falla frágil o servir para simular el comportamiento elástico antes de la plasticidad en la falla dúctil.

5.4. Modelización de la barra 1D, caso dúctil

A partir del planteamiento matemático que se ha hecho en capítulos anteriores, es decir, considerando ablandamiento y utilizando la aproximación de discontinuidad débil se plantea la siguiente clase GradientLayer():

```
class GradientLayer(tf.keras.layers.Layer):
    def __init__(self, model, **kwargs):
        self.model = model
        super().__init__(**kwargs)
    def call(self, x, E, D, eps_l, sigma_y, L, h):
        with tf.GradientTape() as g:
            g.watch(x)
            with tf.GradientTape() as gg:
                gg.watch(x)
                u = self.model(x)
                eps = gg.gradient(u, x)
                condition = tf.math.logical_and(tf.math.greater(x, L/2-h),
                tf.math.less(x, L/2+h))
                sigma=tf.where(condition, D*(eps-eps_l)+sigma_y, E*eps)
            ds_dx=g.gradient(sigma, x)
        return u, eps, sigma, ds_dx
```

En este caso la relación constitutiva incluye la situación de ablandamiento en la banda definida por el parámetro h . El módulo network presenta las siguientes líneas:

```
# POTENCIAL DE ENERGÍA
```

```

u_1 = 0.5*(sigma**2/self.E) + 0.5*self.H*(eps-sigma/self.E)**2

# CONDICIÓN INICIAL OUTPUT

u_2 = u_model(x_2)

# CONDICIÓN DE CONTORNO OUTPUT

u_3 = (u_model(x_3) - self.delta)

```

En estas se calcula la función de densidad de energía de la ecuación 34, que será minimizada. Este código se presenta en el Anexo C. La red se implementa con las siguientes características:

- Número de puntos de muestra (Collocation Points): 50000
- Número de épocas: 30000 en la primera iteración, 10000 en las demás.
- Callbacks: No se implementan callbacks
- Capas ocultas de la red neuronal: 4 capas ocultas de 10, 30, 40, 30 y 10 neuronas respectivamente.
- Función de activación: Tangente hiperbólica

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (39)$$

Con este modelo de PINN se obtiene los resultados de las siguientes ilustraciones:

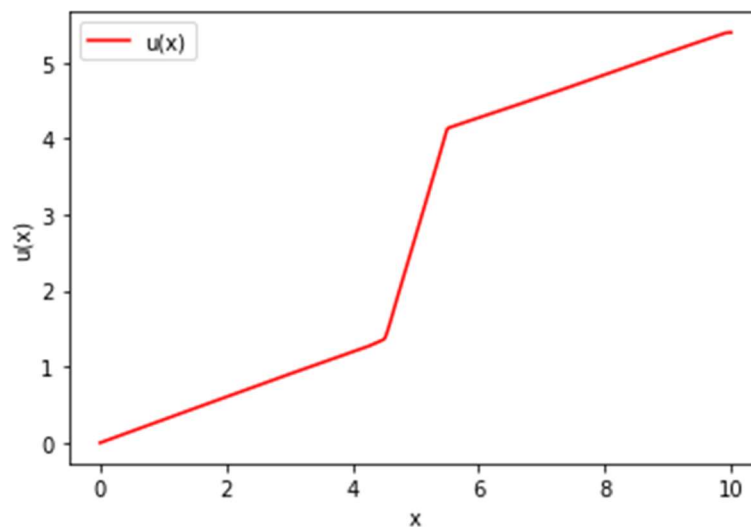


Ilustración 20.- Resultado de desplazamiento para caso dúctil (d=5.4)

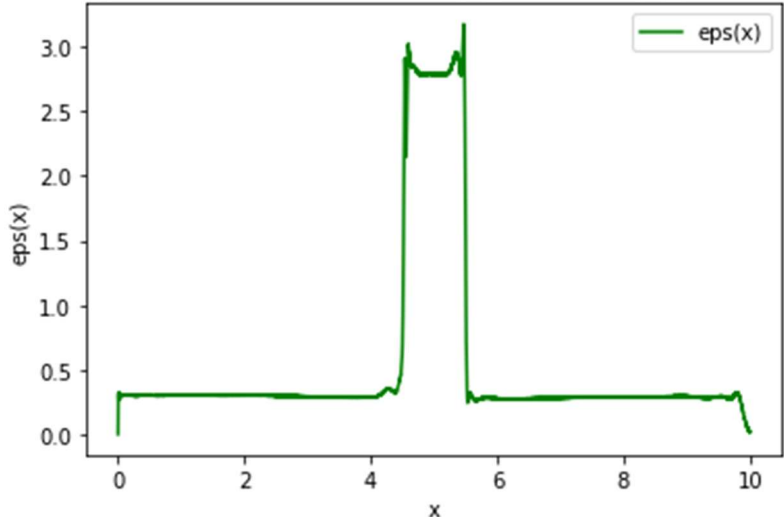


Ilustración 21.- Resultado de deformación para caso dúctil (d=5.4)

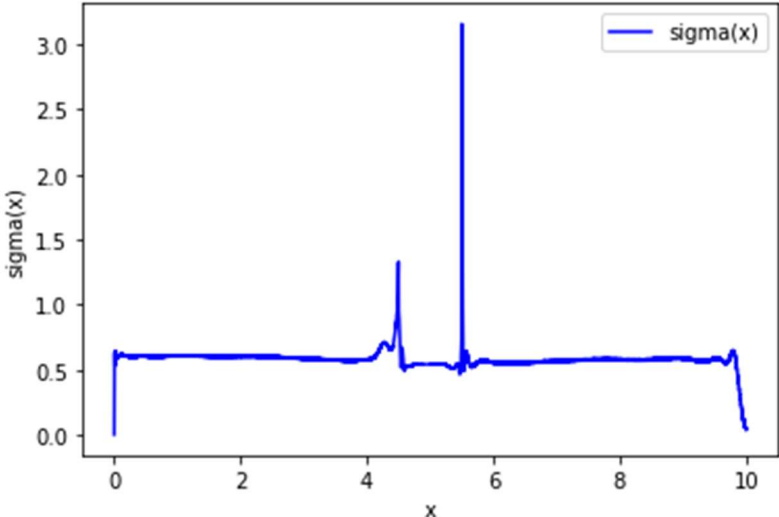


Ilustración 22.- Resultado de tensión para caso dúctil (d=5.4)

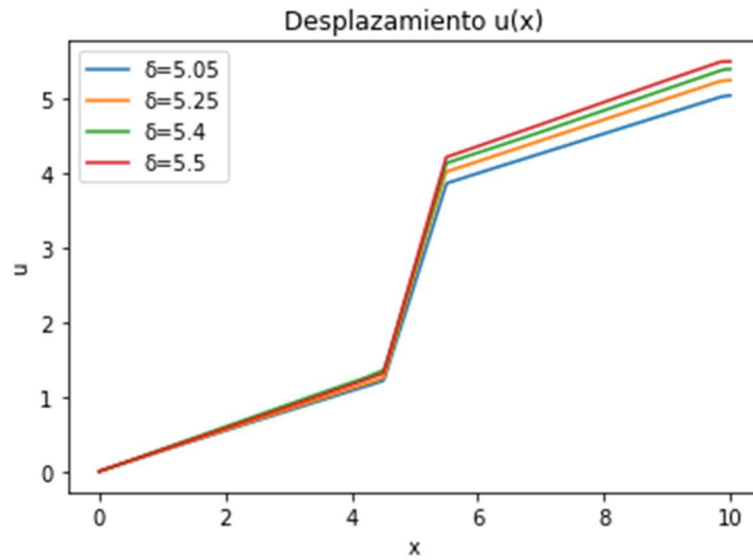


Ilustración 23.- Resultados de desplazamiento, caso dúctil, para distintos desplazamientos en el extremo

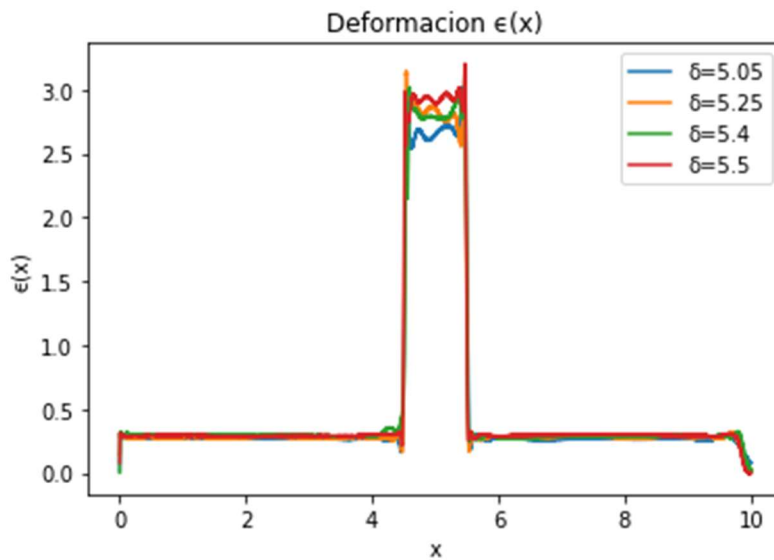


Ilustración 24.- Resultados de deformación, caso dúctil, para distintos desplazamientos en el extremo

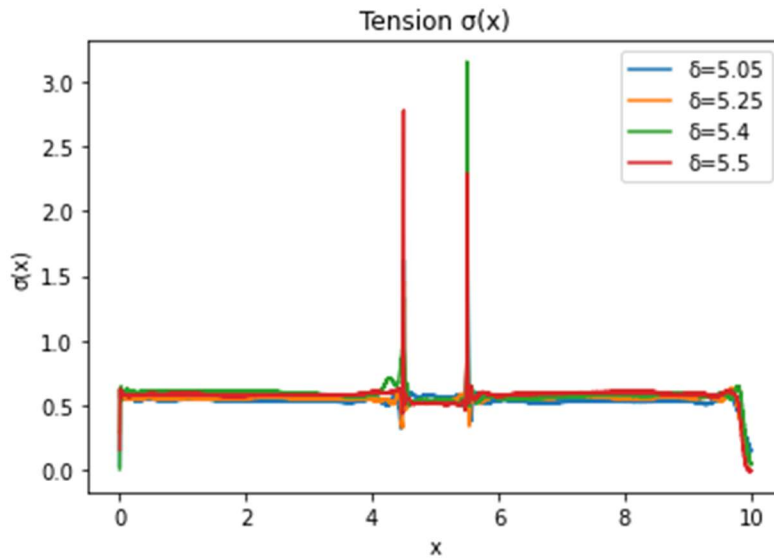


Ilustración 25.- Resultados de tensión, caso dúctil, para distintos desplazamientos en el extremo

Como se observa en las ilustraciones 20-22 tal como se plantea en el código la PINN es capaz de capturar el comportamiento elastoplástico de la barra, generando una banda de localización. Sin embargo, también es evidente que, tal como está planteado el código, se tiende a designar la totalidad de la deformación a la zona plástica inclusive para casos de carga leve (ilustraciones 23-25), esta situación impide ver una evolución en la carga.

Para hacer frente a este problema se plantea otro modelo de PINN inspirado en la separación de componentes del desplazamiento que se encuentra en la literatura (Manzoli et al., 1999). Esta descomposición del desplazamiento considera una parte regular y una componente de salto en la zona de la banda de localización, tal como se muestra en la ilustración 26.

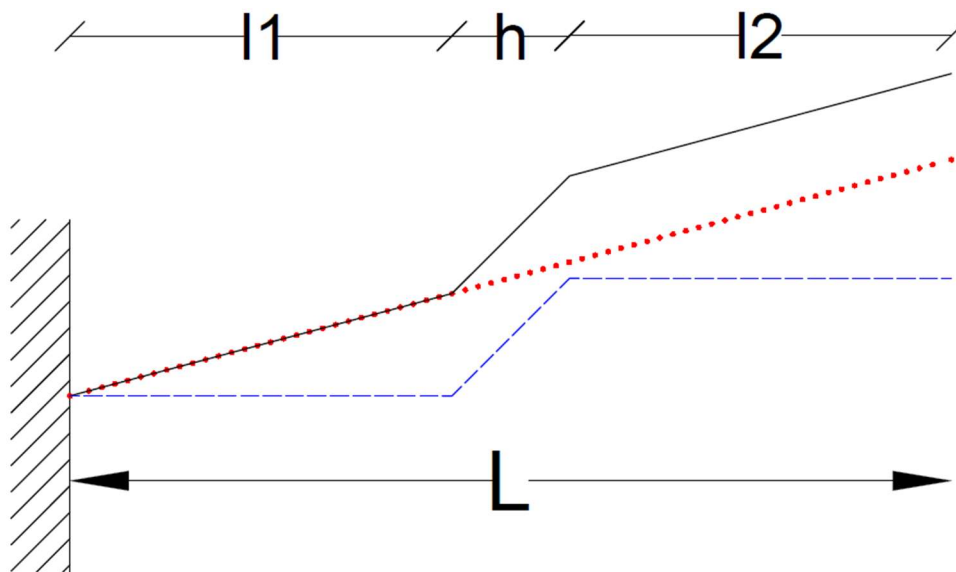


Ilustración 26.- Descomposición del desplazamiento. En rojo la componente regular y en azul la componente del salto.

Considerando lo anterior y partiendo del hecho que la componente regular tiene un comportamiento absolutamente lineal, similar al caso de falla frágil, se puede construir un nuevo modelo de red como el que sigue:

```
def build(self, num_inputs=1, num_outputs=1):
    # CAPA INPUT
    input1 = tf.keras.layers.Input(shape=(num_inputs,))
    x1 = input1
    x2 = input1

    #CAPAS OUTPUT
    out1=tf.keras.layers.Dense(num_outputs, kernel_initializer='ones',
    use_bias=False)(x1)

    banda = CustomActivationLayer(self.L, self.h)
    out2 = banda(x2)

    out = tf.math.add(out1,out2)

    u_model = tf.keras.models.Model(inputs=input1, outputs=[out, out1,
    out2])
```

En el código anterior se presenta la forma de crear la red neuronal principal, como se observa se opta por carecer de capas ocultas, esto ya que una parte del problema, el desplazamiento regular, es una función lineal. Para la componente de salto del desplazamiento se ha generado una función de activación propia definida por:

$$\sigma(x) = \begin{cases} 0 & \text{si } x < \frac{L}{2} - \frac{h}{2} \\ 1 & \text{si } x > \frac{L}{2} + \frac{h}{2} \\ \frac{1}{h} \left(x - \frac{L-h}{2} \right) & \text{caso contrario} \end{cases} \quad (40)$$

Esta función de activación se aplica en la capa CustomActivationLayer(), el código completo se puede revisar en el Anexo D, esta capa además cuenta con un parámetro entrenable k que cumple la función de coeficiente para determinar el valor del salto al multiplicarlo por la función de activación. De esta manera el modelo de red neuronal solo presenta dos parámetros entrenables, un peso w y un coeficiente de k, cuya interpretación física es la de deformación elástica y deformación plástica respectivamente.

La red neuronal principal se entrena a través de la PINN propiamente dicha, esta se presenta en las siguientes líneas de código:

```
t1=self.L*0.5*self.E*eps_e**2
t2=2*self.h*0.5*(((self.eps_l**2+self.sigma_y**2)**0.5+(eps_e**2+(self.D*(eps_t-self.eps_l)+self.sigma_y)**2)**0.5)*(eps_p*np.sin(self.theta)))

u_1 = t1+t2

u_2 = tf.square(u_model(x_2)[1])

# CONDICIÓN DE CONTORNO OUTPUT

u_3 = 10*tf.square(tf.math.subtract(u_model(x_3)[0],self.delta))
```

En estas líneas el término u_1 considera la energía total de la barra, t1 representa la energía almacenada y t2 la energía disipada, tal como se presenta en la ecuación 19. Con este modelo se obtiene los resultados de las siguientes ilustraciones:

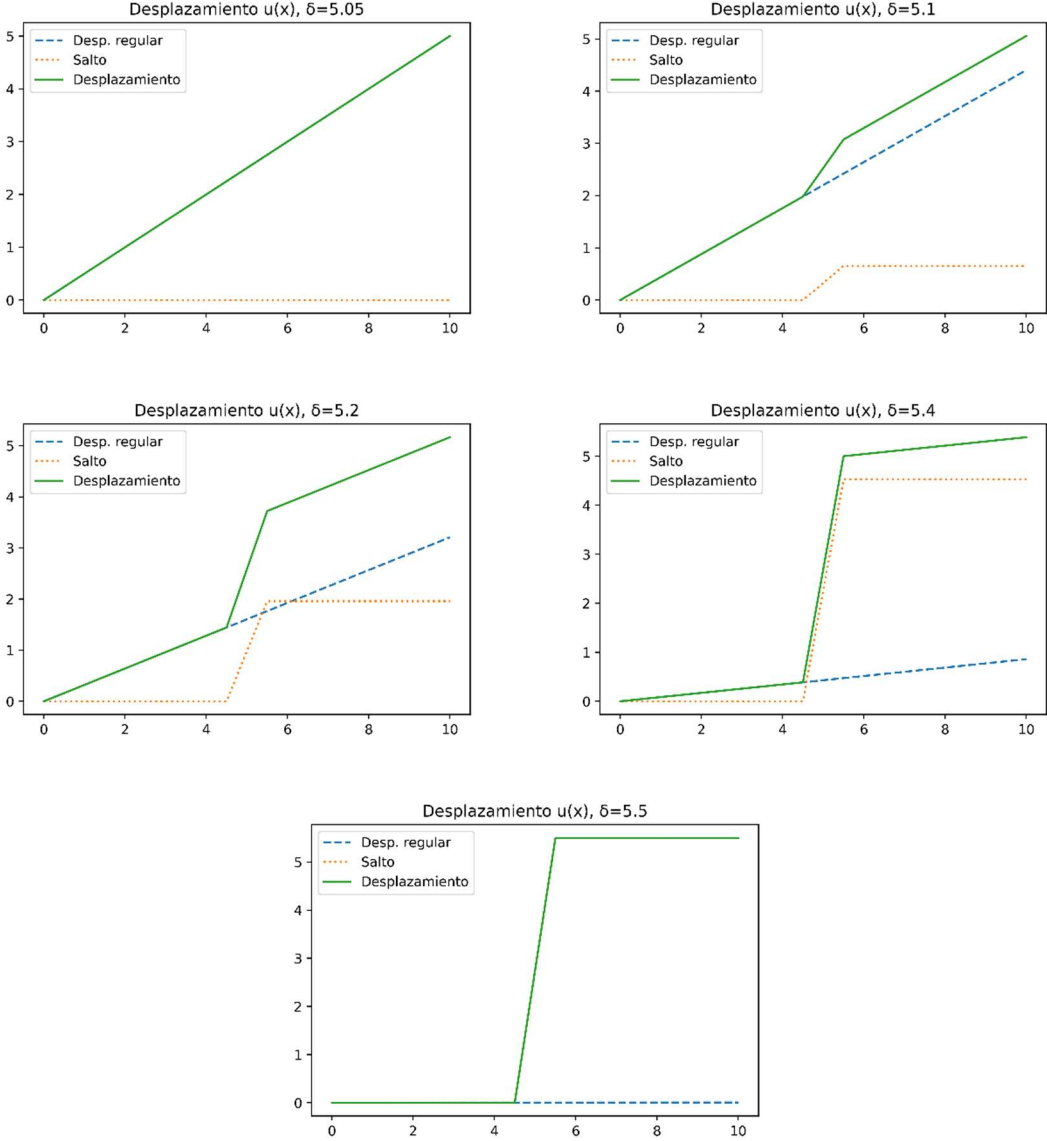


Ilustración 27.- Desplazamiento y sus componentes regular y salto para distintos desplazamientos en el extremo

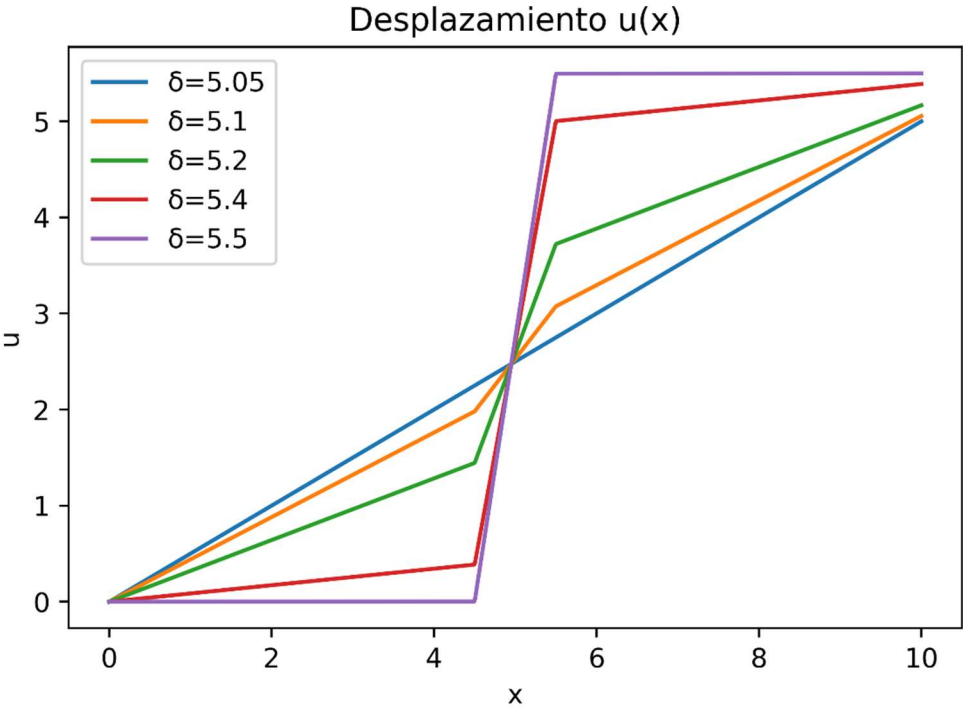


Ilustración 28.- Resultado de desplazamiento caso dúctil.

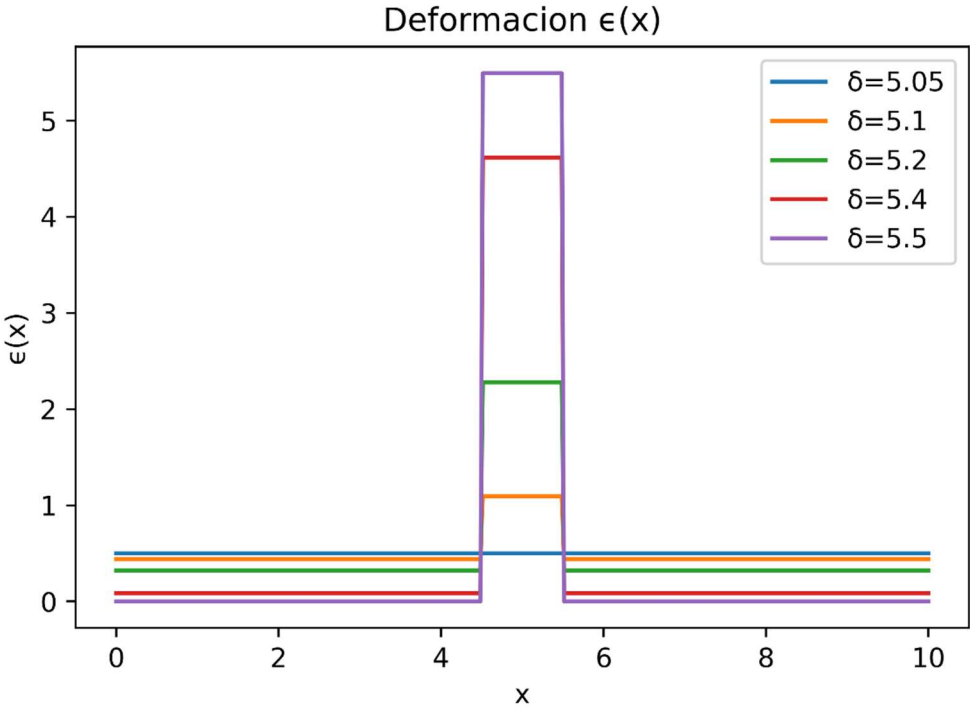


Ilustración 29.- Resultado de deformación caso dúctil.

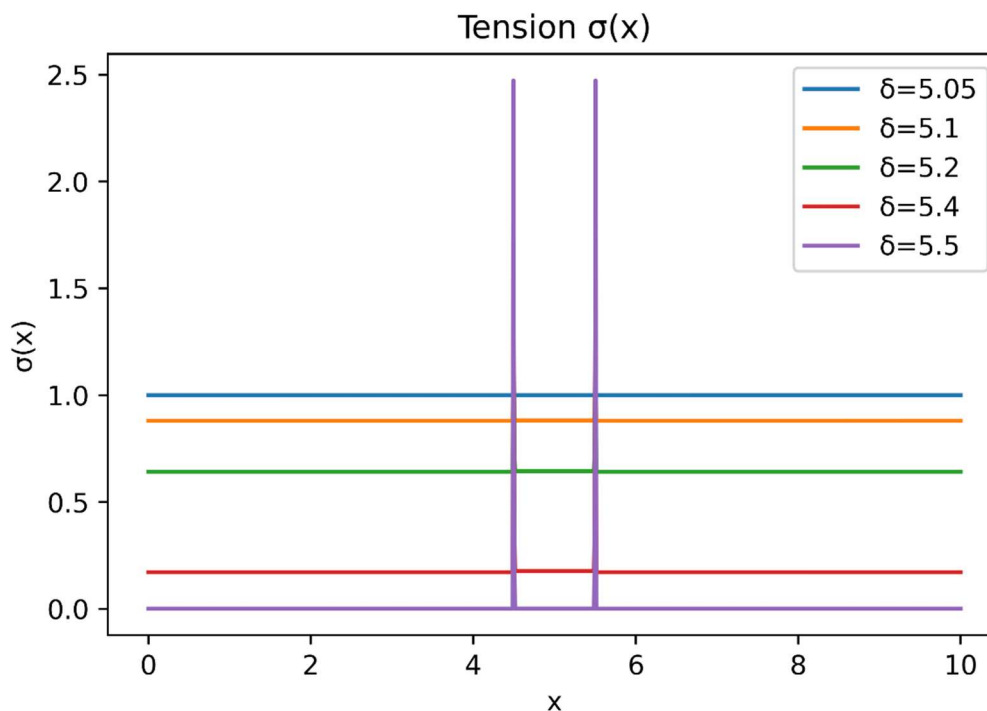


Ilustración 30.- Resultado de tensión caso dúctil.

6. Discusión y Conclusiones

A lo largo de la sección 5 se puede observar la implementación progresiva de la PINN partiendo desde su aplicación a la modelización del mecanismo más sencillo, este es la falla frágil considerando únicamente elasticidad, hasta la consideración del caso elastoplástico principal objeto del presente trabajo.

En lo referente al comportamiento elástico es de señalar que el no utilizar capas ocultas en una red neuronal, además de deberse a la sencillez del problema, también obedece a que, por la naturaleza lineal del fenómeno, el uso de capas intermedias con funciones de activación (ReLU, tanh, etc.) impondría un comportamiento no lineal y en realidad interferiría con la solución en vez de ayudar a obtenerla.

Bajo el mismo razonamiento, también se hace uso de únicamente una neurona de entrada y una de salida para tener únicamente dos parámetros entrenables (un peso w y un sesgo b) que conformen la ecuación de una recta ya que es conocida la naturaleza lineal del comportamiento elástico.

En relación a la modelización del mecanismo de falla dúctil, tema principal de este trabajo, se puede observar que la modelización matemática empleando el enfoque variacional que se presenta en la ecuación 18 y que ha sido deducida de manera equivalente en la ecuación 19 permite la resolución de problemas de localización de deformaciones.

Al implementar estas ecuaciones en la PINN elaborada del apartado 5.4 se puede observar el resultado en las ilustraciones 20-25. Comparando este resultado con el de la solución analítica del apartado 5.1 se observa claramente el comportamiento inelástico, es decir, la localización de deformaciones, en la zona comprendida por la banda de localización.

Este resultado demuestra la capacidad de la red neuronal para capturar el comportamiento de régimen inelástico y aún más su capacidad de modelar mecanismos de falla. Aun así se puede observar que para la primera PINN implementada el resultado no es lo suficientemente preciso en lo referente a la evolución del estado de carga, la primera implementación parece estar concentrando toda la deformación a la zona de la banda de localización desde el inicio de su formación, es decir, inmediatamente después se ha superado el umbral elástico y además de converger siempre hacia un mismo resultado lo cual indica una falencia en la captura del comportamiento elastoplástico.

Este hecho puede deberse a la propia naturaleza de la red neuronal y del espacio de solución que genera, este espacio atípico puede generar pérdida de convexidad en la función de pérdida analizada, provocando la aparición de mínimos locales que 'oculten' la solución real del problema, más aún en una implementación compleja que considera varias capas ocultas de varias neuronas cada una. Aun así, se requiere una investigación a profundidad para comprender la razón subyacente por la cual este tipo de implementación de PINN falla.

Para superar este problema se elaboró el segundo modelo de PINN que captura de mucha mejor manera el resultado al estar mejor adaptado a la sencillez del problema. Inspirado en la descomposición del desplazamiento y a lo realizado en el caso de falla frágil se puede generar una red sin capas ocultas, 1 neurona de entrada y 2 de salida, lo que mejora el desempeño al ser una implementación más sencilla, la única adaptación necesaria fue la implementación de una función de activación propia (ecuación 40).

Este hecho, el poder implementar de dos maneras distintas la PINN, y en ambas capturar el comportamiento elastoplástico, demuestra una característica esencial del método de redes neuronales y es su versatilidad, ya que puede ser modificado de diversas maneras para obtener resultados más precisos, y en algunos casos, como el de este trabajo, las modificaciones pasan por simplificar el problema.

Es importante notar que en ambas implementaciones de la red neuronal se utilizó únicamente el principio variacional como medio para determinar la localización, esto resulta novedoso ya que no implica la implementación de herramientas adicionales tales como phase-fields, ampliamente utilizados en la literatura.

Esta implicación abre una nueva línea de investigación en el uso de redes neuronales para su aplicación en problemas relacionados con la mecánica de sólidos. Es de señalar que el problema analizado, el de la barra unidimensional, dado su sencillez es posible resolverlo de manera analítica o con otro método más extendido como es el de los elementos finitos, inclusive, para este caso en particular, la solución a través de estos dos métodos es mucho más rápida y eficiente.

Sin embargo, es de resaltar lo novedoso del método, así como los resultados obtenidos a partir de este análisis exploratorio. Las PINN vistas de esta manera constituyen una nueva herramienta aun en etapa de exploración por lo que es necesario aun profundizar más en su aplicación a diversos problemas, tales como los que se plantean en la mecánica de sólidos. Los resultados obtenidos en el presente trabajo constituyen un punto de partida para el análisis y la profundización en el estudio de la naturaleza y aplicación de las PINN al fenómeno de localización de deformaciones.

Referencias

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., ... Zheng, X. (2016). TensorFlow: A System for Large-Scale Machine Learning. *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, 265–283.
- Alessi, R. (2013). *Variational Approach to Fracture Mechanics with Plasticity* [Ecole Polytechnique X]. <https://pastel.archives-ouvertes.fr/pastel-00847970>
- Arfken, G. B., Griffing, D. F., Kelly, D. C., & Priest, J. (1984). MOTION IN ONE DIMENSION. In *International Edition University Physics* (pp. 62–85). Elsevier. <https://doi.org/10.1016/B978-0-12-059858-8.50009-1>
- Borja, R. I. (2013). *Plasticity*. Springer Berlin Heidelberg. <https://doi.org/10.1007/978-3-642-38547-6>
- Bourdin, B., Francfort, G. A., & Marigo, J.-J. (2008). The Variational Approach to Fracture. *Journal of Elasticity*, 91(1), 5–148. <https://doi.org/10.1007/s10659-007-9107-3>
- Chen, Y., McFarland, D. M., Wang, Z., Spencer, B. F., & Bergman, L. A. (2010). Analysis of Tall Buildings with Damped Outriggers. *Journal of Structural Engineering*, 136(11), 1435–1443. [https://doi.org/10.1061/\(ASCE\)ST.1943-541X.0000247](https://doi.org/10.1061/(ASCE)ST.1943-541X.0000247)
- Clarke, J. (2014). *Early Structural Steel in London Buildings: A discreet revolution*. Historic England.
- Colored neural network. (2019, May 7). https://commons.wikimedia.org/wiki/File:Colored_neural_network_es.svg
- Cotterell, B. (2002). The past, present, and future of fracture mechanics. *Engineering Fracture Mechanics*, 69(5), 533–553. [https://doi.org/10.1016/S0013-7944\(01\)00101-1](https://doi.org/10.1016/S0013-7944(01)00101-1)
- Francfort, G. A., & Marigo, J.-J. (1998). Revisiting brittle fracture as an energy minimization problem. *Journal of the Mechanics and Physics of Solids*, 46(8), 1319–1342. [https://doi.org/10.1016/S0022-5096\(98\)00034-9](https://doi.org/10.1016/S0022-5096(98)00034-9)
- Ganchenkova, M., & Nieminen, R. M. (2010). Mechanical Properties of Silicon Microstructures. *Handbook of Silicon Based MEMS Materials and Technologies*, 179–219. <https://doi.org/10.1016/B978-0-8155-1594-4.00011-5>

- Goswami, S., Anitescu, C., Chakraborty, S., & Rabczuk, T. (2020). Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. *Theoretical and Applied Fracture Mechanics*, 106. <https://doi.org/10.1016/j.tafmec.2019.102447>
- Griewank, A., & Walther, A. (2008). *Evaluating Derivatives*. Society for Industrial and Applied Mathematics. <https://doi.org/10.1137/1.9780898717761>
- Griffith, A. A. (1921). VI. The phenomena of rupture and flow in solids. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 221(582–593), 163–198. <https://doi.org/10.1098/rsta.1921.0006>
- Griffith, A. A. (1924). The theory of rupture. *Proceedings of the 1st International Congress of Applied Mechanics*, 55–63.
- Haghighat, E., Raissi, M., Moure, A., Gomez, H., & Juanes, R. (2021). A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379, 113741. <https://doi.org/10.1016/J.CMA.2021.113741>
- Hahn, B. H., & Valentine, D. T. (2013). Introduction to Numerical Methods. In *Essential Matlab for Engineers and Scientists* (pp. 301–328). Elsevier. <https://doi.org/10.1016/B978-0-12-394398-9.00014-9>
- Ho, L., & Fatahi, B. (2016). One-Dimensional Consolidation Analysis of Unsaturated Soils Subjected to Time-Dependent Loading. *International Journal of Geomechanics*, 16(2). [https://doi.org/10.1061/\(ASCE\)GM.1943-5622.0000504](https://doi.org/10.1061/(ASCE)GM.1943-5622.0000504)
- Hussein, E. M. A. (2007). COLLISION KINEMATICS. In *Radiation Mechanics* (pp. 67–151). Elsevier. <https://doi.org/10.1016/B978-008045053-7/50003-3>
- Irwin, G. R. (1948). Fracture dynamics. *Fracturing of Metals*, 147–166.
- Katsikis, D., Muradova, A. D., & Stavroulakis, G. E. (2022). A Gentle Introduction to Physics-Informed Neural Networks, with Applications in Static Rod and Beam Problems. *Journal of Advances in Applied & Computational Mathematics*, 9, 103–128. <https://doi.org/10.15377/2409-5761.2022.09.8>

- Kingma, D. P., & Ba, J. L. (2014). Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. <https://arxiv.org/abs/1412.6980v9>
- Kirkaldy, D. (1866). *Results of an experimental inquiry into the tensile strength and other properties of various kinds of wrought-iron and steel*. David Kirkaldy.
- Kosky, P., Balmer, R., Keat, W., & Wise, G. (2021). Civil Engineering. *Exploring Engineering*, 147–180. <https://doi.org/10.1016/B978-0-12-815073-3.00008-9>
- Lanczos, C. (1962). *The Variational Principles of Mechanics*. University of Toronto Press. <http://www.jstor.org/stable/10.3138/j.ctvfrxpd0>
- Lemaitre, J. (1992). *A Course on Damage Mechanics*. Springer Berlin Heidelberg. <https://doi.org/10.1007/978-3-662-02761-5>
- Man, C., & Tsai, C. W. (2007). Stochastic Partial Differential Equation-Based Model for Suspended Sediment Transport in Surface Water Flows. *Journal of Engineering Mechanics*, 133(4), 422–430. [https://doi.org/10.1061/\(ASCE\)0733-9399\(2007\)133:4\(422\)](https://doi.org/10.1061/(ASCE)0733-9399(2007)133:4(422))
- Manzoli, O. L., Oliver, J., & Cervera, M. (1999). *Localizacion de deformaciones: Analisis y Simulacion Numerica de Discontinuidades en Mecanica de Solidos*.
- Maugin, G. A., & Muschik, W. (1994). Thermodynamics with Internal Variables. Part II. Applications. *Journal of Non-Equilibrium Thermodynamics*, 19(3). <https://doi.org/10.1515/jnet.1994.19.3.250>
- Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2018). *Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations*. <https://www.sciencedirect.com/science/article/pii/S0021999118307125>
- Rao, S. S. (2001). THEORY OF VIBRATION | Variational Methods. *Encyclopedia of Vibration*, 1344–1360. <https://doi.org/10.1006/RWVB.2001.0119>
- Rice, J. (1976). Localization of plastic deformation. *Theoretical and Applied Mechanics*.
- Ritchie, R. O., & Liu, D. (2021). Introduction to Fracture Mechanics. In *Introduction to Fracture Mechanics* (1st ed.). Elsevier. <https://doi.org/10.1016/C2020-0-03038-0>

- Rodríguez Cedillo, P. S., & Ulloa Vanegas, J. I. (2015). *Enfoque energético a la modelización de fractura no lineal unidimensional* [Tesis de Grado, Universidad de Cuenca]. <http://dspace.ucuenca.edu.ec/handle/123456789/22876>
- Rodríguez, P., Ulloa, J., Samaniego, C., & Samaniego, E. (2018). A variational approach to the phase field modeling of brittle and ductile fracture. *International Journal of Mechanical Sciences*, 144, 502–517. <https://doi.org/10.1016/j.ijmecsci.2018.05.009>
- Rossmannith, H. P. (1997). The importance of engineering fracture mechanics in structural integrity: A short history of fracture mechanics. *Technology, Law and Insurance*, 2(4), 195–229. <https://doi.org/10.1080/135993797349786>
- Rubenstein, D. A., Yin, W., & Frame, M. D. (2022). Fundamentals of fluid mechanics. *Biofluid Mechanics*, 17–70. <https://doi.org/10.1016/B978-0-12-818034-1.00002-5>
- Rudnicki, J. W., & Rice, J. R. (1975). Conditions for the localization of deformation in pressure-sensitive dilatant materials. *Journal of the Mechanics and Physics of Solids*, 23(6), 371–394. [https://doi.org/10.1016/0022-5096\(75\)90001-0](https://doi.org/10.1016/0022-5096(75)90001-0)
- Samaniego, E., Anitescu, C., Goswami, S., Nguyen-Thanh, V. M., Guo, H., Hamdia, K., Zhuang, X., & Rabczuk, T. (2020). An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. *Computer Methods in Applied Mechanics and Engineering*, 362. <https://doi.org/10.1016/j.cma.2019.112790>
- Sancaktar, E. (2019). Mechanics of solids and structures. *ASME International Mechanical Engineering Congress and Exposition, Proceedings, 10 PART A*, 53–90. <https://doi.org/10.1016/B978-0-12-818283-3.00002-6>
- Simo, J. C., & Hughes, T. J. R. (1998). *Computational Inelasticity* (Vol. 7). Springer-Verlag. <https://doi.org/10.1007/b98904>
- Simo, J. C., Oliver, J., & Armero, F. (1993). An analysis of strong discontinuities induced by strain-softening in rate-independent inelastic solids. *Computational Mechanics*, 12(5), 277–296. <https://doi.org/10.1007/BF00372173/METRICS>
- Ulloa, J., Rodríguez, P., & Samaniego, E. (2016). On the modeling of dissipative mechanisms in a ductile softening bar. *Journal of Mechanics of Materials and Structures*, 11(4), 463–490. <https://doi.org/10.2140/jomms.2016.11.463>

- Vadyala, S. R., Betgeri, S. N., Matthews, J. C., & Matthews, E. (2022). A review of physics-based machine learning in civil engineering. *Results in Engineering*, 13. <https://doi.org/10.1016/j.rineng.2021.100316>
- Wieghardt, K., Sommerfeld, A., & Rossmannith, H. P. (1995). On splitting and cracking of elastic bodies. *Fatigue & Fracture of Engineering Materials and Structures*, 18(12), 1371–1405. <https://doi.org/10.1111/j.1460-2695.1995.tb00864.x>
- Woolf, B. P. (2009). Machine Learning. In *Building Intelligent Interactive Tutors* (pp. 221–297). Elsevier. <https://doi.org/10.1016/B978-0-12-373594-2.00007-1>
- Zehnder, A. T. (2012). *Fracture Mechanics* (2012th ed., Vol. 62). Springer Netherlands. <https://doi.org/10.1007/978-94-007-2595-9>
- Zehnder, A. T. (2013). Griffith Theory of Fracture. In Q. J. Wang & Y.-W. Chung (Eds.), *Encyclopedia of Tribology* (pp. 1570–1573). Springer US. https://doi.org/10.1007/978-0-387-92897-5_259

Anexos

Anexo A-Solución Analítica:

```

def SOLUCION_ANALITICA(E,D,L,h,delta,sigma_y):
    l1=(L-h)/2 #Se considera la banda entre (l-h)/2,(l+h)/2
    l2=l1
    eps_l=sigma_y/E
    DESP=[]
    DEFOR=[]
    TENSION=[]
    FUERZA=[]
    for i in range(len(delta)):
        eps=delta[i]/L
        if eps<=eps_l:
            ...
            Para valores que aún no superan
            el límite elástico se calcula solo la zona
            elástica
            ...
            x=[0,l1,l1+h,l1+h+l2]
            x1=np.linspace(0,L,100)
            u=[0,l1*delta[i]/L,(l1+h)*delta[i]/L,delta[i]]
            defor=[eps for j in range(len(x1))]
            tension=[E*eps for j in range(len(x1))]
            DESP.append(u)
            DEFOR.append(defor)
            TENSION.append(tension)

```

```
F=np.mean(tension)
FUERZA.append(F)
else:
    '''
    Para valores que superan el límite
    elástico se calcula la deformación
    plástica
    '''
    A=np.array([[1,1,1],
                [E/l1,-D/h,0],
                [1/l1,0,-1/l2]])

    b=np.array([[delta[i]],
                [sigma_y-D*eps_l],
                [0]])

    desp=np.linalg.solve(A,b)

    eps_s=desp[1,0]/h
    eps_r1=desp[0,0]/l1
    eps_r2=desp[2,0]/l2

    sigma_r1=E*eps_r1
    sigma_r2=E*eps_r2
    sigma_s=D*(eps_s-eps_l)+sigma_y
```

```
x=[0, l1, l1+h, l1+h+l2]

u=[0, desp[0,0], desp[0,0]+desp[1,0], desp[0,0]+desp[1,0]+desp[2,0]]

x1=np.linspace(0,L,100)
defor=np.where((x1 > l1) & (x1 < (l1+h)),eps_s,eps_r1)
x2=np.linspace(0,L,100)
tension=np.where((x2 > l1) & (x2 <
(l1+h)),sigma_s,sigma_r1)

if np.mean(tension)<=0:
    ...

    La tensión no puede ser menor a cero
    ...

    print("ROTURA DEL MATERIAL")
    return x,x1,DESP,DEFOR,TENSION,FUERZA

break

DESP.append(u)
DEFOR.append(defor)
TENSION.append(tension)

F=np.mean(tension)
FUERZA.append(F)
```

```
    return x,x1,DESP,DEFOR,TENSION,FUERZA

E=2
D=-0.2
L=10
h=1
delta=[i/10 for i in range(50,56,1)]
sigma_y=1
SOL=SOLUCION_ANALITICA(E, D, L, h, delta, sigma_y)
```


Anexo B-Código PINN Caso Frágil:

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import os

from tensorflow.keras.callbacks import TensorBoard
import datetime

from keras.callbacks import Callback
'''
LAYER
'''

class GradientLayer(tf.keras.layers.Layer):
    def __init__(self, model, **kwargs):
        self.model = model
        super().__init__(**kwargs)

    def call(self, x, E):
        with tf.GradientTape() as g:
            g.watch(x)
            with tf.GradientTape() as gg:
                gg.watch(x)
                u = self.model(x)
                eps = gg.gradient(u, x)
                sigma = E*eps
                if tf.math.reduce_mean(sigma)>sigma_y:
                    sigma=0*eps
            ds_dx =g.gradient(sigma, x)
```

```
        return u, eps, sigma, ds_dx
'''
NETWORK
'''
class PINN:
    def __init__(self,E, L, delta, cond):
        self.E=E
        self.L=L
        self.delta=delta
        self.cond=cond

    def build(self, num_inputs=1, layers=[1], activation='relu',
num_outputs=1):
        inputs = tf.keras.layers.Input(shape=(num_inputs,))
        x = inputs
        out = tf.keras.layers.Dense(num_outputs,
kernel_initializer='he_normal')(x)
        u_model = tf.keras.models.Model(inputs=inputs, outputs=out)
        grads=GradientLayer(model=u_model)
        x_1 = tf.keras.layers.Input(shape=(1,))
        x_2 = tf.keras.layers.Input(shape=(1,))
        x_3 = tf.keras.layers.Input(shape=(1,))
        u, eps, sigma, ds_dx = grads(x_1, E=self.E)
        # POTENCIAL DE ENERGIA
        u_1 = 0.5*self.E*eps**2
        # CONDICION INICIAL OUTPUT
        u_2 = 100*u_model(x_2)
        # CONDICION DE CONTORNO OUTPUT
```

```

    u_3 = 100*(u_model(x_3) - self.delta)

    # MODELO PINN PARA LA ECUACION DE LA BARRA

    return u_model, tf.keras.models.Model(

        inputs=[x_1, x_2, x_3] , outputs=[u_1, u_2, u_3])

class stopAtLossValue(Callback):

    def __init__(self, eps_l,E):

        super().__init__()

        self.eps_l = eps_l

        self.E=E

        self.threshold = 0.5*self.E*self.eps_l**2

    def on_train_end(self, logs=None):

        u_1_pred, _, _ = self.model.predict(x_train)

        if np.mean(u_1_pred) > self.threshold:

            print('SIGMA MAYOR AL MAXIMO, SE PRODUCE FRACTURA')

            self.model.stop_training = True

'''

MAIN

'''

E = 2

L = 10

sigma_y = 1

delta=[0.25,0.5,1,2,3,4,5,5.05]

eps_l=sigma_y/E

num_train_samples=1000

num_test_samples=1000

```

```
x_1 =
np.linspace(0,L,num_train_samples).reshape((num_train_samples,1))
x_2 = np.zeros((num_train_samples, 1))
x_3 = L*np.ones((num_train_samples, 1))
u_1 = np.zeros((num_train_samples, 1))
u_2 = np.zeros((num_train_samples, 1))
u_3 = np.zeros((num_train_samples, 1))
x_train = [x_1, x_2, x_3]
y_train_target = [u_1, u_2, u_3]
u_pred=[]
tension=[]
deformacion=[]
for d in delta:
    print('CODIGO ELASTICO')
    if d==delta[0]:
        cond=0
        u_model, PINN_model = PINN(E, L, d, cond).build()
        print(u_model.summary())
        print(PINN_model.summary())
        PINN_model.compile(optimizer='adam',
                            loss=tf.keras.losses.mse,
                            metrics=[tf.keras.metrics.mse],
                            )
        model_history = PINN_model.fit(x_train, y_train_target,
batch_size=num_train_samples, epochs=10000, verbose=1,
callbacks=[stopAtLossValue(eps_l, E)])
```

```
PINN_model.save_weights("training_FRAGIL1/cp.ckpt")
u_model.save_weights("training_FRAGIL2/cp.ckpt")
x_test = np.linspace(0, L, num_test_samples)
u = u_model.predict(x_test, batch_size=num_train_samples)
u1,u2,u3 = PINN_model.predict([x_1,x_2,x_3],
batch_size=num_train_samples)
x_test=x_test.flatten()
u=u.flatten()
eps=np.gradient(u,x_test)
sigma=(2*u1*E)**0.5
deformacion.append(eps)
tension.append(sigma)
else:
cond=1
u_model, PINN_model = PINN(E, L, d, cond).build()
print(u_model.summary())
print(PINN_model.summary())
PINN_model.compile(optimizer='adam',
                    loss=tf.keras.losses.mse,
                    metrics=[tf.keras.metrics.mse],
                    )
PINN_model.load_weights("training_FRAGIL1/cp.ckpt")
u_model.load_weights("training_FRAGIL2/cp.ckpt")
model_history = PINN_model.fit(x_train, y_train_target,
batch_size=num_train_samples, epochs=1000, verbose=1,
callbacks=[stopAtLossValue(eps_l, E)])
```

```
PINN_model.save_weights("training_FRAGIL1/cp.ckpt")
u_model.save_weights("training_FRAGIL2/cp.ckpt")
x_test = np.linspace(0, L, num_test_samples)
u = u_model.predict(x_test, batch_size=num_train_samples)
u1,u2,u3 = PINN_model.predict([x_1,x_2,x_3],
batch_size=num_train_samples)
x_test=x_test.flatten()
u=u.flatten()
eps=np.gradient(u,x_test)
sigma=(2*u1*E)**0.5
deformacion.append(eps)
tension.append(sigma)
```

Anexo C – Código PINN Caso Dúctil, Implementación 1

```
import os

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

import tensorflow as tf

tf.compat.v1.logging.set_verbosity(tf.compat.v1.logging.ERROR)

import numpy as np

import matplotlib.pyplot as plt

from keras.callbacks import Callback

from time import time

'''

LAYER

'''

class GradientLayer(tf.keras.layers.Layer):

    def __init__(self, model, **kwargs):

        self.model = model

        super().__init__(**kwargs)

    def call(self, x, E, D, eps_l, sigma_y, L, h):

        with tf.GradientTape() as g:

            g.watch(x)

            with tf.GradientTape() as gg:

                gg.watch(x)

                u = self.model(x)

                eps = gg.gradient(u, x)

                condition = tf.math.logical_and(tf.math.greater(x, L/2-h), tf.math.less(x, L/2+h))

                sigma=tf.where(condition,D*(eps-eps_l)+sigma_y, E*eps)
```

```

        deps_dx=g.gradient(eps, x)

        return u, eps, sigma, deps_dx
'''
PINN
'''

class PINN:

    def __init__(self, E, D, eps_l, sigma_y, L, delta,h):

        self.E = E

        self.D = D

        self.eps_l = eps_l

        self.sigma_y = sigma_y

        self.L = L

        self.delta=delta

        self.h=h

        self.H=E*D/(E-D)

    def build(self, num_inputs=1, layers=[10,30,40,30,10],
activation='tanh', num_outputs=1):

        # CAPA INPUT

        inputs = tf.keras.layers.Input(shape=(num_inputs,))

        # CAPAS OCULTAS

        x = inputs

        for layer in layers:

            x = tf.keras.layers.Dense(layer, activation=activation,
                                       kernel_initializer='he_normal')(x)

        # CAPA OUTPUT

        out = tf.keras.layers.Dense(num_outputs,

```



```

kernel_initializer='he_normal')(x)

    u_model = tf.keras.models.Model(inputs=inputs, outputs=out)
    grads = GradientLayer(u_model)

    # INPUT ECUACION: x_1 -> random
    x_1 = tf.keras.layers.Input(shape=(1,))

    # CONDICION INICIAL INPUT: x_2 =0
    # CONDICION DE CONTORNO INPUT: x_3 = L
    x_2 = tf.keras.layers.Input(shape=(1,))
    x_3 = tf.keras.layers.Input(shape=(1,))

    # CALCULAR GRADIENTES
    u, eps, sigma, ds_dx = grads(x_1, E=self.E, D = self.D,
eps_l=self.eps_l, sigma_y=self.sigma_y, L=self.L, h=self.h)

    # POTENCIAL DE ENERGIA
    u_1 = 0.5*(sigma**2/self.E)+0.5*self.H*(eps-sigma/self.E)**2

    # CONDICION INICIAL OUTPUT
    u_2 = u_model(x_2)

    # CONDICION DE CONTORNO OUTPUT
    u_3 = (u_model(x_3) - self.delta)

    # MODELO PINN PARA LA ECUACIÓN DE LA BARRA
    return u_model, tf.keras.models.Model(
        inputs=[x_1, x_2, x_3] , outputs=[u_1, u_2, u_3])
'''

MAIN
'''

if __name__ == '__main__':

```

```
start = time()

# NÚMERO DE MUESTRAS DE ENTRENAMIENTO (COLLOCATION POINTS)
num_train_samples = 50000

# NÚMERO DE MUESTRAS DE TESTEO (PARA GRAFICAR)
num_test_samples = 50000

# PARAMETROS FISICO Y GEOMETRICOS

E = 2

D = -0.2

eps_l = 0.5

sigma_y = 1

delta = [5.05,5.25,5.4,5.5]

L = 10

h = 0.5

# CREAR INPUT DE ENTRENAMIENTO

x_1 = np.linspace(0,L,num_train_samples).reshape((num_train_samples,1))
x_2 = np.zeros((num_train_samples, 1))
x_3 = L*np.ones((num_train_samples, 1))

# CREAR OUTPUT OBJETIVO

u_1 = np.zeros((num_train_samples, 1))
u_2 = np.zeros((num_train_samples, 1))
u_3 = np.zeros((num_train_samples, 1))
```

```
x_train = [x_1, x_2, x_3]
y_train_target = [u_1, u_2, u_3]
u_pred=[]
tension=[]
deformacion=[]
i=0
for d in delta:
    u_model, PINN_model = PINN(E, D, eps_l, sigma_y, L, d,
h).build()
    print(u_model.summary())
    print(PINN_model.summary())
    PINN_model.compile(optimizer='adam',
                        loss=tf.keras.losses.mae,
                        metrics=[tf.keras.metrics.mae],
                        )
    if i!=0:
        PINN_model.load_weights("training_DUCTIL_PINN/cp.ckpt")
        u_model.load_weights("training_DUCTIL_umodel/cp.ckpt")
        model_history = PINN_model.fit(x_train, y_train_target,
batch_size=num_train_samples, epochs=10000)
    else:
        model_history = PINN_model.fit(x_train, y_train_target,
batch_size=num_train_samples, epochs=30000)
    print(f"TOTAL TRAINING TIME = {round((time() - start), 2)}
seconds")
```

```
PINN_model.save_weights("training_DUCTIL_PINN/cp.ckpt")
u_model.save_weights("training_DUCTIL_u_model/cp.ckpt")
x_test = x_1
u = u_model.predict(x_test, batch_size=num_train_samples)
PINN_ar = PINN_model.predict([x_1,x_2,x_3],batch_size=num_train_samples)
x_test=x_test.flatten()
u=u.flatten()
eps=np.gradient(u,x_test)
x=tf.constant(x_test)
eps=tf.constant(eps)
condition = tf.math.logical_and(tf.math.greater(x,L/2-h),
tf.math.less(x, L/2+h))
sigma=tf.where(condition,D*(eps-eps_l)+sigma_y, E*eps)
sigma=tf.make_ndarray(tf.make_tensor_proto(sigma))
ds_dx=np.gradient(sigma,x_test)
u_pred.append(u)
deformacion.append(eps)
tension.append(sigma)
```

Anexo D – Código PINN Caso Dúctil, Implementación 2

```

'''
LAYER
'''

class GradientLayer(tf.keras.layers.Layer):
    def __init__(self, model, **kwargs):
        self.model = model
        super().__init__(**kwargs)

    def call(self, x, E, D, eps_l, sigma_y, L, h):
        with tf.GradientTape(persistent=True) as g:
            g.watch(x)
            with tf.GradientTape(persistent=True) as gg:
                gg.watch(x)
                u, ur, uS = self.model(x)
                eps_e = gg.gradient(ur, x)
                eps_p = gg.gradient(uS, x)
                eps_t = gg.gradient(u, x)
                condition = tf.math.logical_and(tf.math.greater(x, L/2-h),
                tf.math.less(x, L/2+h))
                sigma=tf.where(condition,D*(eps_t-eps_l)+sigma_y,
                E*eps_t)
                sigma_e=E*eps_e
                sigma_p=D*(eps_t-eps_l)+sigma_y
            deps_e=g.gradient(eps_e, x)
            deps_p=g.gradient(eps_p, x)
            deps_t=g.gradient(eps_t, x)

```

```
        return u, ur, uS, eps_e, eps_p, eps_t, sigma, deps_e,
        deps_p, deps_t

class CustomActivationLayer(tf.keras.layers.Layer):

    def __init__(self, L, h):

        super(CustomActivationLayer, self).__init__()

        self.L = L

        self.h = h

    def build(self, input_shape):

        self.K = self.add_weight(shape=(), initializer='ones',
        trainable=True, name='k')

        super(CustomActivationLayer, self).build(input_shape)

    def call(self, inputs):

        x = inputs

        threshold = self.L / 2

        lower_bound = threshold - self.h

        upper_bound = threshold + self.h

        condition =
        tf.math.logical_and(tf.math.greater(x, lower_bound), tf.math.less(x,
        upper_bound))

        condition_low = tf.math.less_equal(x, lower_bound)

        condition_high = tf.math.greater_equal(x, upper_bound)

        activation = tf.where(condition_low, 0.0, x)

        activation = tf.where(condition_high, 1.0 , activation)

        activation = tf.where(condition, (x - lower_bound) /
        (upper_bound - lower_bound), activation)

        return activation * self.K
```

```
'''  
PINN  
'''  
  
class PINN:  
    def __init__(self, E, D, eps_l, sigma_y, L, delta,h,c):  
        self.E = E  
        self.D = D  
        self.eps_l = eps_l  
        self.sigma_y = sigma_y  
        self.L = L  
        self.delta=delta  
        self.h=h  
        self.H=E*D/(E-D)  
        self.theta=np.arctan(E)  
        self.c=c  
  
    def build(self, num_inputs=1, num_outputs=1):  
        # CAPA INPUT  
        input1 = tf.keras.layers.Input(shape=(num_inputs,))  
        x1 = input1  
        x2 = input1  
  
        #CAPAS OUTPUT  
        out1 = tf.keras.layers.Dense(num_outputs,  
kernel_initializer='ones', use_bias=False)(x1)  
        banda = CustomActivationLayer(self.L, self.h)  
        out2 = banda(x2)
```

```

    out = tf.math.add(out1,out2)

    u_model = tf.keras.models.Model(inputs=input1, outputs=[out,
out1, out2])

    grads = GradientLayer(u_model)

    # INPUT ECUACION: x_1 -> random
    x_1 = tf.keras.layers.Input(shape=(1,))
    x_2 = tf.keras.layers.Input(shape=(1,))
    x_3 = tf.keras.layers.Input(shape=(1,))

    # CALCULAR GRADIENTES

    u, ur, uS, eps_e, eps_p, eps_t, sigma, deps_e, deps_p,
deps_t = grads(x_1, E=self.E, D = self.D, eps_l=self.eps_l,
sigma_y=self.sigma_y, L=self.L, h=self.h)

    eps_e=tf.reduce_mean(eps_e)

    eps_p=tf.reduce_max(eps_p)

    eps_t=eps_e+eps_p

    t1=self.L*0.5*self.E*eps_e**2
t2=2*self.h*0.5*(((self.eps_l**2+self.sigma_y**2)**0.5+(eps_e**2+(se
lf.D*(eps_t-
self.eps_l)+self.sigma_y)**2)**0.5)*(eps_p*np.sin(self.theta)))

    u_1 = t1+t2

    u_2 = tf.square(u_model(x_2)[1])

    # CONDICIÓN DE CONTORNO OUTPUT

    u_3 =
10*tf.square(tf.math.subtract(u_model(x_3)[0],self.delta))

    # MODELO PINN PARA LA ECUACION DE LA BARRA

    return u_model, tf.keras.models.Model(

        inputs=[x_1, x_2, x_3] , outputs=[u_1,u_2,u_3])
...

```

MAIN


```
'''  
  
if __name__ == '__main__':  
    start = time()  
  
    # NÚMERO DE MUESTRAS DE ENTRENAMIENTO (COLLOCATION POINTS)  
    num_train_samples = 1000  
  
    # NÚMERO DE MUESTRAS DE TESTEO (PARA GRAFICAR)  
    num_test_samples = 1000  
  
    # PARÁMETROS FÍSICOS Y GEOMÉTRICOS  
  
    E = 2  
  
    D = -0.2  
  
    eps_l = 0.5  
  
    sigma_y = 1  
  
    delta = [5.05, 5.1, 5.2, 5.4, 5.5]  
  
    c = 0.001  
  
    L = 10  
  
    h = 0.5  
  
    # CREAR INPUT DE ENTRENAMIENTO  
  
    x_1 =  
np.linspace(0,L,num_train_samples).reshape((num_train_samples,1))  
  
    x_2 = np.zeros((num_train_samples, 1)) #  
x_2 = 0  
  
    x_3 = L*np.ones((num_train_samples, 1))  
  
    u_1 = np.zeros((num_train_samples, 1)) # u_1 = 0  
  
    u_2 = np.zeros((num_train_samples, 1)) # u_2 = 0  
  
    u_3 = np.zeros((num_train_samples, 1)) # u_3 = 0  
  
    x_train = [x_1, x_2, x_3]  
  
    y_train_target = [u_1, u_2, u_3]
```

```
u_pred=[]
tension=[]
deformacion=[]
for d in delta:
    u_model, PINN_model = PINN(E, D, eps_l, sigma_y, L, d, h,
c).build()
    print(u_model.summary())
    print(PINN_model.summary())

PINN_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=
0.001),
                    loss=tf.keras.losses.mae,
                    metrics=[tf.keras.metrics.mae],
                    )

    model_history = PINN_model.fit(x_train, y_train_target,
batch_size=num_train_samples, epochs=20000)

    u, ur, us = u_model.predict(x_1,
batch_size=num_train_samples)

    PINN_ar =
PINN_model.predict([x_1,x_2,x_3],batch_size=num_train_samples)
```