

# Nombre Fancy

David Arturo Rodríguez Herrera

Diciembre 2020

## Índice

<b>1. Resumen</b>	<b>3</b>
<b>2. Introducción</b>	<b>3</b>
<b>3. Objetivos</b>	<b>3</b>
<b>4. Marco Teórico</b>	<b>4</b>
4.1. Ecuación Constitutiva . . . . .	4
4.2. Funciones de Atenuación y Difusión . . . . .	5
4.3. Distancia de influencia $Lr$ . . . . .	6
4.4. Factor de corrección $\lambda_0$ . . . . .	7
4.5. Función de Atenuación Modificada (Función 4) . . . . .	8
<b>5. Metodología</b>	<b>10</b>
5.1. Implementación Elasticidad No Local . . . . .	10
5.1.1. Método FEM . . . . .	10
5.1.2. Modificaciones No Locales . . . . .	11
5.1.3. Estructura del Programa . . . . .	13
5.2. Casos de Estudio . . . . .	15
5.3. Validación . . . . .	19
5.4. Variación de Parámetros . . . . .	20
5.4.1. Variaciones Planteadas . . . . .	20
<b>6. Resultados</b>	<b>22</b>
<b>7. Conclusiones</b>	<b>23</b>
<b>8. Anexos</b>	<b>24</b>
<b>Referencias</b>	<b>25</b>

## Índice de figuras

1. Función biexponencial centrada en 0,0 . . . . .	5
2. Función cónica centrada en 0,0 . . . . .	6
3. Función campana centrada en 0,0 . . . . .	6
4. Funciones de atenuación . . . . .	8

5.	Función de atenuación modificada . . . . .	9
6.	Cambios en la inclusión de elementos no locales . . . . .	11
7.	Estructura de guardado de archivos para las matrices . . . . .	15
8.	Diagrama de clases UML para el programa . . . . .	15
9.	Modelo de placa a tensión, tomado de Pisano <i>et al.</i> (2009) . . . . .	16
10.	Importar librerías y enmallado . . . . .	16
11.	Enmallado de 100 elementos para dominio cuadrado . . . . .	17
12.	Condiciones de borde integración y solución . . . . .	17
13.	Post proceso . . . . .	18
14.	Resultados gráficos para un enmallado de 100 elementos . . . . .	18
15.	Modelo de barra unidimensional tomado de Pisano <i>et al.</i> (2009) . . . . .	18
16.	Perfiles de $\varepsilon_x$ en el caso de estudio 1 . . . . .	20
17.	Diagrama de flujo del proceso iterativo . . . . .	21
18.	Gráficas de validación . . . . .	24

## Índice de Tablas

1.	Estructura del archivo de enmallado . . . . .	14
----	---	----

Resumen

Introducción

Objetivos

## Marco Teórico

### Ecuación Constitutiva

Las bases de la teoría no local las sentó Kröner (1967) en su artículo «*Elasticity theory of materials with long range cohesive forces*». En este, Kröner expone que la energía sobre un cristal cúbico primitivo se debe en parte a las fuerzas de corto alcance y en parte a las fuerzas de largo alcance, de tal manera, Kröner llega a la siguiente ecuación:

$$E = \frac{1}{2} \left[ \int_V C_{ijkl} \varepsilon_{ij}(\mathbf{x}) \varepsilon_{kl}(\mathbf{x}) dv \right] + \frac{1}{2} \left[ \int_V \int_{V'} C_{ijkl}(\mathbf{x}) \varepsilon_{ij}(\mathbf{x}) \varepsilon_{kl}(\mathbf{x}') dv' dv \right] \quad (1)$$

Por ello, Kröner adopta el término No local para referirse a la interacción de fuerzas de largo alcance que se representan en el segundo término de la parte derecha de la ecuación.

Con ello, Kröner llegó a la siguiente ecuación constitutiva:

$$\sigma_{ij}(\mathbf{x}) = C_{ijkl} \varepsilon_{kl}(\mathbf{x}) + \int_{V'} C_{ijkl}(|\mathbf{x} - \mathbf{x}'|) \varepsilon_{kl}(\mathbf{x}') dv' \quad (2)$$

En esta ecuación, el segundo término representa el aporte no local mientras que el primer término representa el aporte local clásico. Además, se introduce el tensor  $C_{ijkl}(|\mathbf{x} - \mathbf{x}'|)$  llamado el tensor elástico material. Este tensor depende de la distancia entre el punto de interés y los puntos del dominio. En este punto, no existía una manera sencilla de calcular  $C_{ijkl}(|\mathbf{x} - \mathbf{x}'|)$ .

No fue hasta 5 años después que Eringen y Edelen (1972) desarrollaron un modelo matemático donde se formula de manera completa la elasticidad no local. De este trabajo nace la teoría lineal en la que Eringen (1987) describe los dos modelos de elasticidad no local y sus aplicaciones. Los dos modelos son el modelo integral, el cual describe la interacción de fuerzas a larga distancia, y el modelo diferencial que describe interacciones a corta distancia (Polizzotto, 2001). Eringen muestra una gran variedad de problemas que se pueden solucionar con la teoría no local, entre los que destacan la propagación de ondas y el uso de esta teoría en el estudio de fractura y grietas. Esto se debe a que en las puntas de las grietas, la elasticidad local presenta singularidades, obteniendo así un esfuerzo infinito, mientras que en la teoría no local hay un aporte del volumen adyacente, lo cual evita tener esfuerzos infinitos (Eringen, 1987).

En el trabajo «Theory of Nonlocal Elasticity and Some Applications.», Eringen expone una manera de calcular el tensor elástico material.

$$C_{ijkl}(|\mathbf{x} - \mathbf{x}'|) = \alpha(|\mathbf{x} - \mathbf{x}'|) C_{ijkl} \quad (3)$$

Donde se introduce  $\alpha$  como la *función de atenuación* (este nombre será modificado posteriormente), por lo que Eringen llega a la siguiente ecuación constitutiva:

$$\sigma_{ij}(\mathbf{x}) = \int_{V'} \alpha(|\mathbf{x} - \mathbf{x}'|) C_{ijkl} \varepsilon_{kl}(\mathbf{x}') dv' \quad (4)$$

A partir de aquí, se resaltan los trabajos de Polizzotto (2001) donde reporta que la ecuación constitutiva se escribe con un factor  $\zeta_1$  y  $\zeta_2$ , donde  $\zeta_1 + \zeta_2 = 1$  con el objetivo de representar el proceso de deformación de un sólido con dos fases. Una fase local ( $\zeta_1$ ) y una fase no local ( $\zeta_2$ ).

$$\sigma_{ij}(\mathbf{x}) = \zeta_1 C_{ijkl} \varepsilon_{kl}(\mathbf{x}) + \zeta_2 \int_{V'} A(|\mathbf{x} - \mathbf{x}'|) C_{ijkl} \varepsilon_{kl}(\mathbf{x}') dv' \quad (5)$$

Donde  $A$  también es llamada función de atenuación pero  $A \neq \alpha$ . Para efectos de este estudio se llamará a la función  $\alpha$  como función de difusión y se llamará a la función  $A$  función de atenuación.

Por último, en la ecuación 4 se tiene dos parámetros locales (Modulo de Young y coeficiente de Poisson) y un parámetro no local (longitud interna  $l$ ), sin embargo, en la ecuación 5 se cuenta con el parámetro adicional  $\zeta_1$ .

## Funciones de Atenuación y Difusión

Las funciones de difusión que expone Eringen (1987) deben cumplir con unos requerimientos mínimos que son:

$$\lim_{l \rightarrow 0} \alpha = 1 \quad (6)$$

$$\alpha(0) = 1 \quad (7)$$

Adicionalmente, para que exista una única solución,  $\alpha > 0$  (Altan, 1989).

Las funciones de atenuación que expone Polizzotto (2001) deben cumplir con unos requerimientos mínimos que son:

$$\lim_{l \rightarrow 0} A = 0 \quad (8)$$

$$\int_{v^\infty} A(|\mathbf{x} - \mathbf{x}'|) dv' = 1 \quad (9)$$

A partir de las ecuaciones 4,5 Polizzotto (2001) establece la siguiente relación:

$$\alpha(\rho) = \zeta_1 A(\rho) + \zeta_1 \delta_{\mathbf{x}-\mathbf{x}'} \quad (10)$$

Donde  $\rho = \frac{r}{l}$  y  $r = |\mathbf{x} - \mathbf{x}'|$ .  $r$  es interpretado comúnmente como la distancia euclidiana.

Como lo enuncia Altan (1989),  $A$  no tiene que ser necesariamente positiva, por lo que se puede elegir de una gran gama de funciones. En este estudio se analizarán 4 funciones de atenuación y su influencia en los resultados obtenidos variando sus parámetros no locales. Las funciones de atenuación son de la forma:

$$A(\rho) = \lambda_0 f(\rho) \quad (11)$$

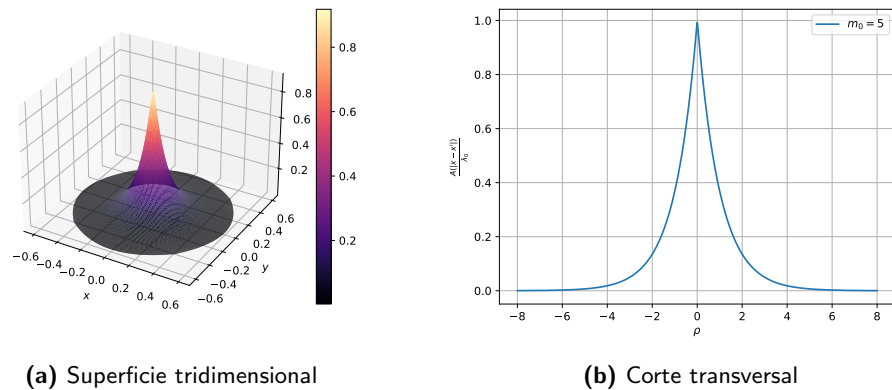
Donde  $\lambda_0$  es un factor de corrección para obligar a cumplir con la condición de  $\int_{v^\infty} A(|\mathbf{x} - \mathbf{x}'|) dv' = 1$

En la literatura se exponen varias funciones de atenuación de las que resaltan:

Nota: el paréntesis  $< f(x) >$  representa el operador de Macauley.

### 1. Función Biexponencial (Función 1)

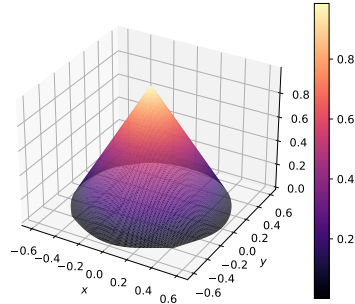
$$A(\rho) = \lambda_0 e^{-\rho} \quad (12)$$



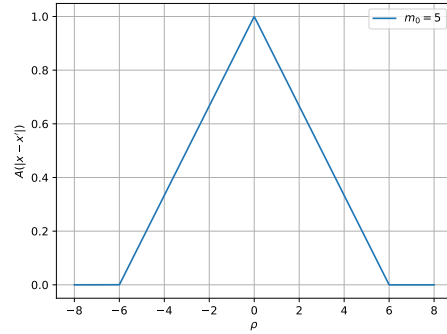
**Figura 1.** Función biexponencial centrada en 0,0

## 2. Función Cónica (Función 2)

$$A(\rho) = \lambda_0 \left\langle 1 - \frac{\rho}{(1 + m_0)} \right\rangle \quad (13)$$



(a) Superficie tridimensional

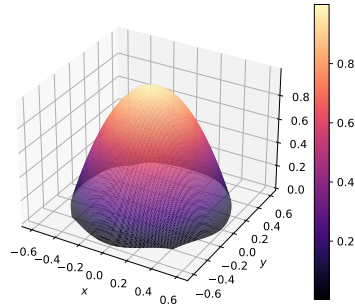


(b) Corte transversal

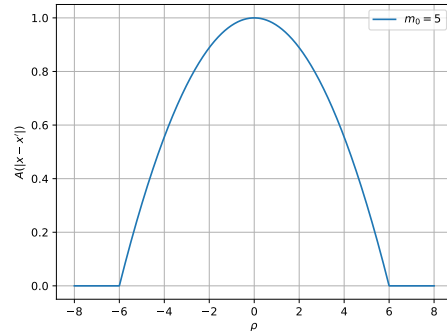
**Figura 2.** Función cónica centrada en 0,0

## 3. Función Campana (Función 3)

$$A(\rho) = \lambda_0 \left\langle 1 - \frac{\rho^2}{(1 + m_0)^2} \right\rangle \quad (14)$$



(a) Superficie tridimensional



(b) Corte transversal

**Figura 3.** Función campana centrada en 0,0

Cada una de estas funciones tiene un identificador (Función i) que será usado posteriormente en el desarrollo de los programas.

Distancia de influencia  $Lr$ 

Si se observa el comportamiento de la figura 1b, se puede apreciar que el valor de la función de atenuación es muy cercana a 0 cuando  $|\rho| \geq 6$ . Es por eso que Polizzotto (2001) define que los efectos no locales solamente se deben tener en cuenta a una distancia máxima  $Lr = 6l$ . Esta afirmación hace que al momento de enmallar un dominio de elementos finitos deba tenerse en cuenta una propiedad material ( $l$ ) para poder modelar el problema no local de manera correcta. Para efectos de este estudio se tomará  $Lr = 6l$ .

Polizzotto (2001) propone un parámetro  $m_0 \geq 1$  donde  $Lr = (1 + m_0)l$ . En las figuras 1b y 1b se usa dicho parámetro. El parámetro obliga a que  $A(\rho) = 0$  cuando  $|\rho| > (1 + m_0)$ . Este comportamiento hace que las funciones de atenuación sean comparables entre sí.

## Factor de corrección $\lambda_0$

Una vez definidas las funciones de atenuación, basta con hallar el factor  $\lambda_0$  para que se cumpla la condición de la ecuación 9.

Dado que  $\lambda_0$  es constante, el proceso para encontrar dicho valor se simplifica. De tal manera, se encuentra el valor de la integral en un dominio circular infinito. Se toma como punto base el origen, ya que, el valor de la integral no depende de la posición base sino de los puntos adyacentes a esta base.

$$t \int_0^{2\pi} \int_0^\infty r A(r/l) dr d\theta = 1 \quad (15)$$

Donde  $r$  es la distancia euclidiana al origen, que en coordenadas polares puede escribirse como el radio.

Note que esta integral se define como integral de área, sin embargo, la integral de la ecuación 9 es una integral volumétrica. Se asume un grosor constante por lo que basta con multiplicar por  $t$ .

Usando la función 1 como referencia se tiene:

$$t \lambda_0 \int_0^{2\pi} \int_0^\infty r e^{-r/l} dr d\theta = 1 \quad (16)$$

$$2\pi \lambda_0 t \int_0^\infty r e^{-r/l} dr = 1 \quad (17)$$

$$2\pi \lambda_0 t \int_0^\infty r e^{-r/l} dr = 1 \quad (18)$$

$$2\pi \lambda_0 t \left( \lim_{r \rightarrow \infty} -e^{-\frac{r}{l}} l^2 - l r e^{-\frac{r}{l}} + l^2 \right) = 1 \quad (19)$$

$$2\pi \lambda_0 l^2 t = 1 \quad (20)$$

Solucionando para  $\lambda_0$ :

$$\lambda_0 = \frac{1}{2\pi l^2 t} \quad (21)$$

Para las funciones 2 y 3 se debe modificar el proceso así:

$$t \int_0^{2\pi} \int_0^\infty r A(r/l) dr d\theta = t \int_0^{2\pi} \int_0^{Lr} r A(r/l) dr d\theta + t \int_0^{2\pi} \int_{Lr}^\infty r A(r/l) dr d\theta \quad (22)$$

Debido a la definición de la función 2 y 3 es evidente que:

$$t \int_0^{2\pi} \int_{Lr}^\infty r A(r/l) dr d\theta = 0 \quad (23)$$

Por lo que el proceso para la función 2 y 3 se simplifica a:

$$t \int_0^{2\pi} \int_0^{Lr} r A(r/l) dr d\theta = 1 \quad (24)$$

Al realizar esta simplificación las funciones de atenuación pierden la propiedad del operador de Macauley, ya que, en este intervalo son positivas.

Resolviendo para ambos casos se obtuvo:

Para la función 2:

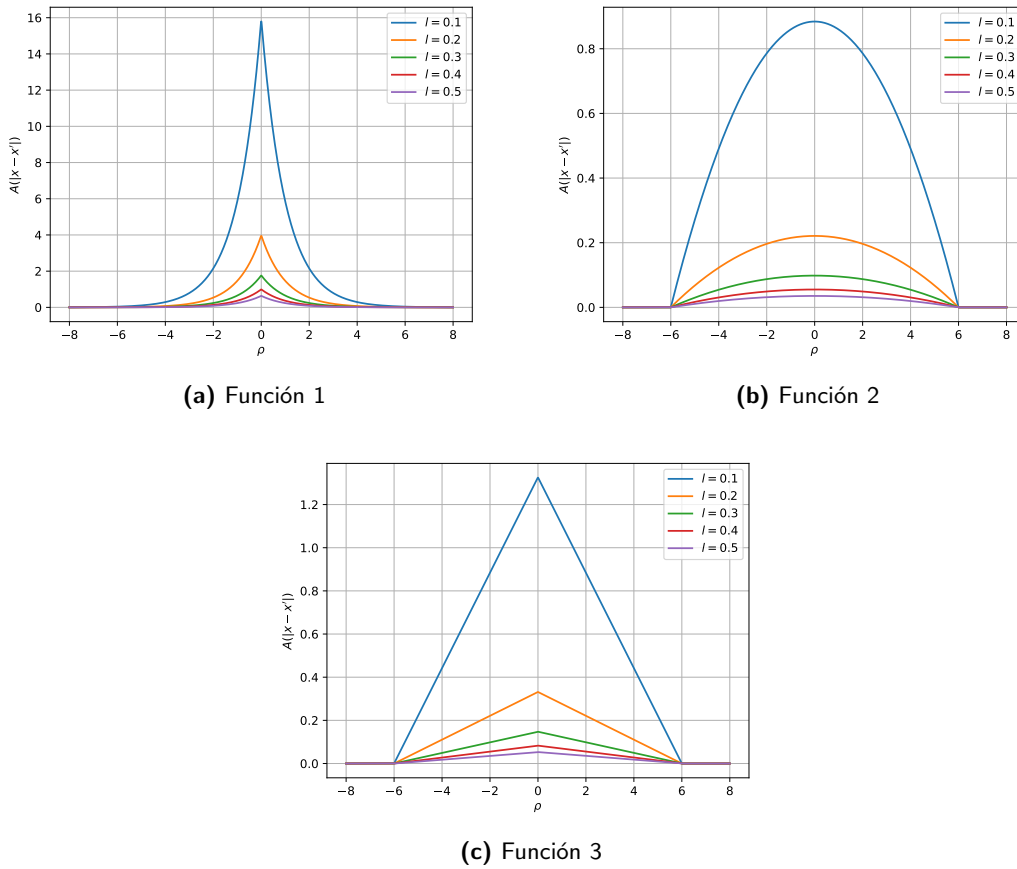
$$\lambda_0 = \frac{3}{2\pi L r^2} \quad (25)$$

$$\lambda_0 = \frac{1}{\pi L r^2} \quad (26)$$

Para la función 3:

Para comprobar que el proceso se realizó correctamente, se realizó la integración numérica de las funciones de atenuación en un dominio circular. En este ensayo se cambiaron los parámetros  $l$  y  $m_0$ . Estos resultados se exponen en la sección 6.

En este punto, las funciones de atenuación están completamente definidas. En la figura 4 se evidencia el comportamiento de dichas funciones cuando varía el parámetro  $l$ .



**Figura 4.** Funciones de atenuación

## Función de Atenuación Modificada (Función 4)

En el artículo «Nonlocal integral elasticity: 2D finite element based solutions» se expone una metodología basada en el método de elementos finitos, el cual adopta el nombre de NL-FEM donde:

$$\hat{K}U = F \quad (27)$$

Donde  $\hat{K}$

$$\hat{K} = \sum_{n=1}^N \left[ \zeta_1 K_n^l + \zeta_2 \sum_{m=1}^M K_{nm}^{nl} \right] \quad (28)$$



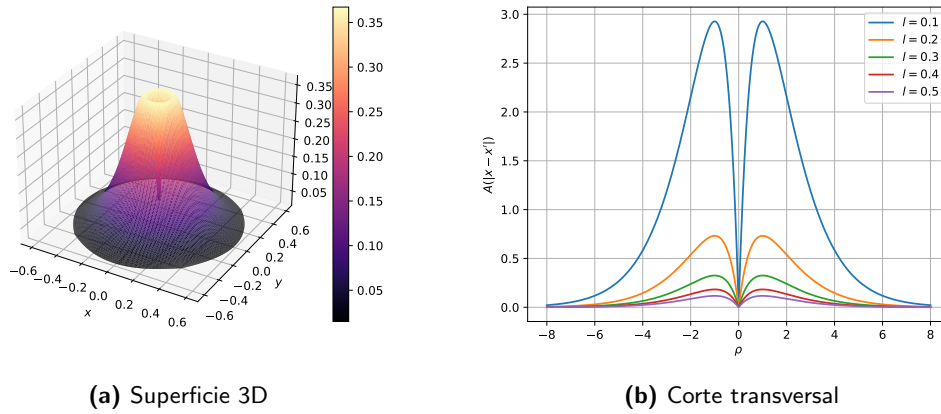
Esto sugiere que para cada elemento  $n$  existe un número  $M$  de elementos no locales  $m$  que se encuentran a una distancia de influencia  $Lr$ . Además, cuando  $n = m$ ,  $\mathbf{K}^{nl}$  se llama matriz de rigidez directa. Siguiendo la ecuación 4 es intuitivo pensar que la matriz de rigidez directa debe ser igual a la matriz de rigidez local, sin embargo, a nivel de elemento no se da esta igualdad, ya que en la integración numérica si existe una distancia entre los puntos de integración. Este efecto hace que:

$$\zeta_1 \mathbf{K}_n^l + \zeta_2 \mathbf{K}_{nn}^{nl} \neq \mathbf{K}_n^l \quad (29)$$

Por ello, se definió una función de atenuación la cual cumpla que  $A(0) = 0$  por lo que se optó por:

$$A(\rho) = \lambda_0 \rho e^{-\rho} \quad (30)$$

Esta función tiene el comportamiento mostrado en la figura 5.



**Figura 5.** Función de atenuación modificada

Adicionalmente se encontró que para que se cumpla la ecuación 9 se necesita:

$$\lambda_0 = \frac{1}{4\pi l^2} \quad (31)$$

## Metodología

En orden de cumplir con los objetivos propuestos en el capítulo 3, se plantea una implementación de NL-FEM. Para ello, se parte de la metodología propuesta en «*Nonlocal integral elasticity: 2D finite element based solutions*» (Pisano *et al.* 2009) que sigue la ecuación constitutiva planteada en el capítulo 4.

## Implementación Elasticidad No Local

Para llegar a la implementación se parte de la implementación clásica (local) de FEM. Para efectos de este estudio se asumen esfuerzos planos, por lo que:

$$\sigma_{xz} = 0 \quad (32a)$$

$$\sigma_{yz} = 0 \quad (32b)$$

$$\sigma_z = 0 \quad (32c)$$

Lo cual permite expresar el tensor de rigidez  $C$  como:

$$C = \begin{pmatrix} C_{11} & C_{12} & 0 \\ C_{12} & C_{11} & 0 \\ 0 & 0 & C_{66} \end{pmatrix} \quad (33)$$

Donde:

$$C_{11} = \frac{E}{1 - \nu^2} \quad (34a)$$

$$C_{12} = \frac{\nu E}{1 - \nu^2} \quad (34b)$$

$$C_{66} = \frac{E}{2(1 + \nu)} \quad (34c)$$

## Método FEM

Se parte de la metodología planteada en Reddy (2005) donde se divide el método en 6 pasos:

### 1. Enmallado:

Se divide el dominio en elementos y se enumeran sus nodos con grados de libertad. Para el caso de elasticidad, se deben tener dos grados de libertad por nodo. El primero pertenece al desplazamiento en  $x$ , comúnmente llamado  $U$  y el segundo pertenece al desplazamiento en  $y$ , comúnmente llamado  $V$ . El número de nodos por elemento depende del orden de aproximación y de la geometría del elemento. Para este estudio se usarán elementos de cuatro lados con 8 nodos por elemento, comúnmente llamados *Serendipity Elements* y elementos triangulares con 6 nodos.

### 2. Ecuaciones a nivel de elemento

Mediante un modelo de elementos finitos (al que se puede llegar con la forma débil de la ecuación diferencial) se realiza el proceso de integración de las matrices de coeficientes, fuerzas nodales y flujos por elemento. Generalmente la integración se realiza usando la cuadratura de Gauss-Legendre.

### 3. Ensamblaje

Se obliga a la solución a ser continua en todo el dominio y se asegura el equilibrio de flujos en los nodos. Esto genera un sistema de ecuaciones el cual se ensambla mediante las matrices y vectores encontrados en el paso anterior.

#### 4. Imposición de condiciones de borde

Se imponen condiciones de borde operando matricialmente los valores conocidos para la solución o asignando directamente el flujo conocido al sistema de ecuaciones.

#### 5. Solución

Se soluciona el sistema de ecuaciones encontrando la solución en cada grado de libertad.

#### 6. Post - Proceso

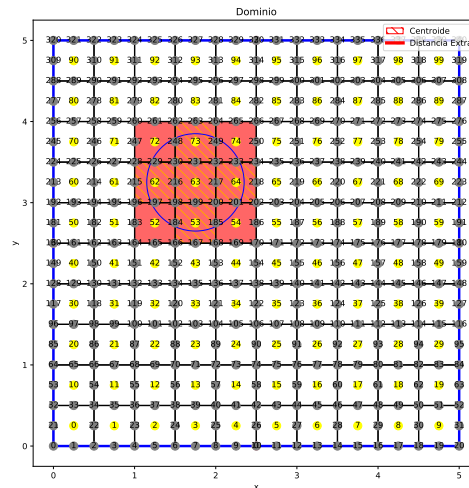
Se calcula el gradiente de la solución (variables secundarias) y se reportan los resultados de manera gráfica.

### Modificaciones No Locales

Para la inclusión de la no localidad se deben modificar los 3 primeros pasos del método clásico. Estos cambios se deben a la inclusión de una integral convolutiva en la ecuación constitutiva y a la elección de la función de atenuación.

#### 1. Enmallado

Uno de los cambios más relevantes en el proceso de NL-FEM es la inclusión de los elementos no locales por cada elemento local. Para ello, la literatura se propone encontrar los elementos cuyos centroides se encuentren dentro de una distancia de influencia  $Lr$  (Zingales *et al.* 2011), sin embargo, Ramirez (2020) sugiere que el proceso de selección de elementos no locales debe tener en cuenta la distancia adicional que se tenga al punto más alejado del elemento. En la figura 6 se muestran gráficamente ambos procesos. Se evalúa el elemento 63. El círculo azul representa un radio de  $Lr = 6 * l$  con  $l = 0.1$  desde el centroide del elemento.



**Figura 6.** Cambios en la inclusión de elementos no locales

Dadas las propiedades de las funciones de atenuación descritas en el capítulo 4, se puede asegurar que si se toma una distancia  $Lr \rightarrow \infty$  el resultado no cambia de manera significativa. En caso de que no se disponga de un algoritmo que reconozca los elementos no locales, esto simplifica el proceso de enmallado. En caso de que se tenga  $Lr \rightarrow \infty$  el costo computacional aumenta severamente, por lo que se aconseja siempre identificar los elementos no locales.

Para este estudio se tomará  $Lr$  variable, pero nunca menor de  $6l$  como se expuso en el capítulo 4.

En conclusión, en un dominio de  $N$  elementos se conoce que: Para cada elemento  $n_i$  existe un número  $M$  de elementos no locales  $m_j$ .  $M$  varía para cada elemento pues depende de su posición en el dominio.

## 2. Ecuaciones a nivel de elemento

Como se expuso en el capítulo 4, se debe realizar una integral convolutiva en el proceso de integración. Por lo tanto, por cada elemento se obtiene una matriz local y un número específico de matrices no locales.

La matriz local se obtiene usando la teoría clásica (local) de elasticidad.

Las matrices no locales se encuentran usando el modelo propuesto por Pisano *et al.* (2009) donde:

$$\mathbf{k}_{nm}^{nloc} = \int_{V_n} \int_{V_m} A(|\mathbf{x} - \mathbf{x}'|/l) \mathbf{B}_n^T(\mathbf{x}) \mathbf{D} \mathbf{B}_n(\mathbf{x}) dV' dV \quad (35)$$

Donde:

$n$  se refiere al elemento local.

$m$  se refiere al elemento no local.

$A$  es la función de atenuación.

$\mathbf{B}_n$  es la matriz de derivadas parciales de las funciones de forma.

$\mathbf{D}$  tensor de rigidez.

$\mathbf{x}$  representa un punto del elemento local  $n$ .

$\mathbf{x}'$  representa un punto del elemento no local  $m$ .

En la terminología de Reddy (2005) se puede reescribir esta ecuación a nivel de elemento separando los desplazamientos en  $x$  y  $y$  en submatrices de la matriz del elemento. Como ejemplo se muestra la matriz  $\mathbf{K}_{uu}$ :

$$\mathbf{U} \mathbf{U}_{nm}^{nloc} [i, j] = t^2 \int_{\Omega_n} \int_{\Omega_m} A(\rho) \left[ C_{11} \frac{\partial \psi_i^l}{\partial x} \frac{\partial \psi_j^{nl}}{\partial x} + C_{66} \frac{\partial \psi_i^l}{\partial y} \frac{\partial \psi_j^{nl}}{\partial y} \right] d\Omega_m d\Omega_n \quad (36)$$

Nótese que a la ecuación 35 es una integral volumétrica, mientras que la ecuación 36 corresponde a una integral de área, por tal razón, se debe multiplicar por el grosor al cuadrado.

Donde:

$\rho$  es la relación  $|\mathbf{x} - \mathbf{x}'|/l$ .

$\psi_i$  es la  $i$ -ésima función de forma.

$C_{11}, C_{66}$  son los componentes del tensor elástico para esfuerzos planos.

El super índice  $l$  significa que la función pertenece al elemento local.

El super índice  $nl$  significa que la función pertenece al elemento no local.

Las funciones están evaluadas en los puntos correspondientes a su elemento, es decir, la función  $\psi_i^{nl}$  está evaluada en los puntos  $x_{nl}, y_{nl}$ .

En los anexos 37 se encuentra el desarrollo de las otras submatrices.

## 3. Ensamblaje

Dado que los elementos no locales tienen aportaciones a la matriz general del sistema, el ensamblaje de dichos elementos debe tenerse en cuenta usando los grados de libertad de los elementos correspondientes. Por ejemplo, si se evalúa la matriz no local del elemento local 3 con respecto al elemento 4, en la matriz del sistema se debe incluir este aporte en las filas que correspondan a los grados de libertad del elemento 3 con las columnas que representen los grados de libertad del elemento 4. Esta generalización hace posible que se puedan tener combinación de diferentes tipos de elementos (como elementos triangulares y rectangulares).

## Estructura del Programa

La implementación computacional propuesta consta de 2 módulos, el módulo procesador y el módulo integrador. La implementación completa se encuentra en [GitHub](#).

### 1. Módulo Procesador

El módulo procesador está escrito en Python y requiere de las librerías *triangle*, *NumPy* y *Matplotlib*. Este módulo se puede encargar de la solución completa de un problema de NL-FEM. Para el enmallado se usa el *wrapper* de *triangle* (Shewchuk, 1996), que lleva el mismo nombre. Fue desarrollado por [Dzhelil Rufat](#) y consta de una serie de funciones que realiza triangulaciones *Delauney*. El enmallado por *triangle* no suele ser uniforme, ya que en el proceso los elementos son refinados para obligar a que los ángulos de los triángulos no sean menores a 35 grados. Adicionalmente Ramirez (2020) desarrolló un algoritmo para realizar el proceso de enmallado en dominios rectangulares identificando elementos no locales a una distancia  $Lr$ . Esto facilitó el proceso de enmallado para los casos de estudio propuestos.

Para las ecuaciones a nivel de elemento se usa la ecuación 36 y sus variaciones en el anexo 37.

Para la integración se usa la cuadratura de Gauss Legendre. Se puede integrar con cualquier número de puntos. Los puntos y pesos se extraen de la librería *NumPy*.

Para el ensamblaje de matrices y vectores se usan las propiedades de los arreglos de *NumPy*.

Para las condiciones de borde se crearon métodos auxiliares que permiten generar las condiciones en los nodos que pertenezcan a segmentos y nodos que se encuentren en puntos (coordenadas) específicos. Los segmentos pueden ser definidos por el usuario o ser generados automáticamente por el programa. Esto se realizó ya que las condiciones de borde generalmente se asignan en los contornos del dominio.

Las condiciones de borde se toman como listas nativas de Python, lo que permite que se puedan manejar de manera sencilla. Un ejemplo es tener condiciones de borde en dos segmentos distintos. Para poder concatenarlas y que el programa reconozca y aplique ambas condiciones adecuadamente basta con usar el símbolo  $+$  entre ambas condiciones de borde y automáticamente el programa será capaz de aplicarlas correctamente. Este proceso de concatenación de condiciones de borde se evidencia en la figura 10. En el caso en el que se aplique una condición de borde natural y esencial al mismo nodo el programa reconocerá la condición de borde esencial.

Para aplicar las condiciones de borde se hace uso del método matricial que cambia las filas y columnas de los grados de libertad con condición de borde a 0 exceptuando la diagonal. En caso de que se apliquen dos condiciones de borde distintas al mismo nodo, el programa reconocerá la segunda condición de borde como la correcta.

Para la solución del sistema de ecuaciones se usa la librería *NumPy* con la función *numpy.linalg.solve*. Esta función usa la librería LAPACK que corre en FORTRAN.

El componente gráfico usa la librería *Matplotlib*. Se pueden graficar las soluciones para las variables principales, así como para las variables secundarias. Para encontrar perfiles en la solución se creó un algoritmo que extrae el perfil interpolando la solución con las funciones de forma. Dado que las funciones de forma se evalúan en el dominio natural, se implementó un algoritmo de mapeo inverso usando el método de Newton.

### 2. Módulo Integrador

Debido a que Python es un lenguaje interpretado (Kuhlman, 2011), se conoce con anterioridad que el lenguaje será lento para procesos iterativos complejos. Para el caso de NL-FEM el paso más costoso computacionalmente es la integración de las matrices no locales, por tal razón, se decidió realizar un segundo módulo que se encargue únicamente de realizar el proceso de integración. Para ello se usó C++ que es un lenguaje compilado. En este programa se siguen exactamente los mismos paradigmas que se introdujeron en el módulo procesador con la única diferencia que al ser un programa ejecutable, se deben proporcionar los parámetros por medio de la consola en lugar de modificarlos en el código fuente.

Gracias al módulo *subprocess* de Python se pudo automatizar el proceso de llamar al programa de C++ desde Python, es decir, el usuario no debe salir de Python para realizar el proceso de integración.

El módulo integrador usa la cuadratura de Gauss Legendre y obtiene los puntos de una implementación propia adaptada de [Rosseta Code](#).

El módulo integrador recibe como parámetros:

- a) Ruta del archivo de enmallado (*string*)
- b) Numero de puntos de Gauss para integrar (*int*)
- c) Módulo de Young (*double*)
- d) Coeficiente de Poisson (*double*)
- e) Grosor (*double*)
- f) longitud interna (*double*)
- g) Nombre de la carpeta donde se quieren guardar las matrices (*string*)
- h) Función de atenuación a usar siguiendo los lineamientos del capítulo 4 (*int*)

### 3. Manejo de datos y recurrencia

Para garantizar la comunicación correcta entre los dos módulos se manejan archivos que guardan la información sobre el enmallado y la integración.

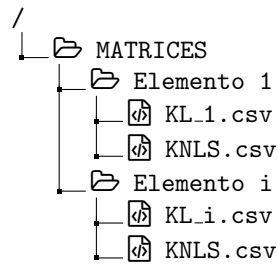
- a) Archivo enmallado: Se parte como base del archivo de enmallado del programa de Ramirez (2020) que sigue la estructura de la tabla 1. Este archivo es usado por el módulo procesador y el módulo integrador.

Al ser el archivo principal, lo ideal es generar los archivos de enmallado previamente a realizar el proceso NL-FEM. Por ello, con el enmallado interno de *triangle* se provee una funcionalidad que exporta el archivo de enmallado con el objetivo que el módulo integrador funcione incluso si no se usa un enmallado previamente definido.

- b) Archivos integración: Dado que generar las matrices de los elementos es un proceso computacionalmente costoso, el módulo integrador generará una serie de carpetas que guardan la información de las matrices. Para cada elemento del enmallado se guarda un archivo *KL.i.csv* que contiene la matriz local del elemento *i*. Para cada elemento del enmallado se guarda un archivo llamado *KNLS.csv* donde cada fila del archivo corresponde a la *j*-ésima matriz no local. Estas matrices se aplanan por columnas y filas por lo que al cargarlas se deben transponer. La estructura del directorio de salida se encuentra en la figura 7.

**Tabla 1.** Estructura del archivo de enmallado

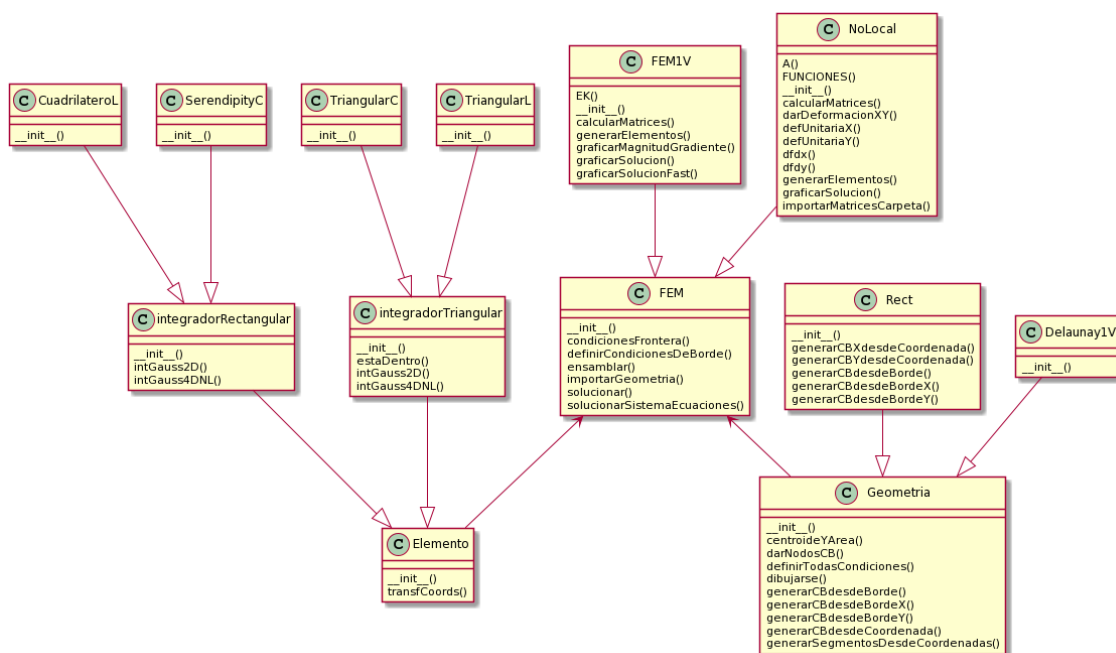
Linea 1	#GDL	#E
$GDL_i$	X	Y
$E_i$	$GDL_1$	$GDL_j$
$ENL_{i,j}$	#ENL	$E_j^{nl}$



**Figura 7.** Estructura de guardado de archivos para las matrices

#### 4. Estructura final

El programa completo se orientó a objetos y su estructura se muestra en el diagrama UML de la figura 8.

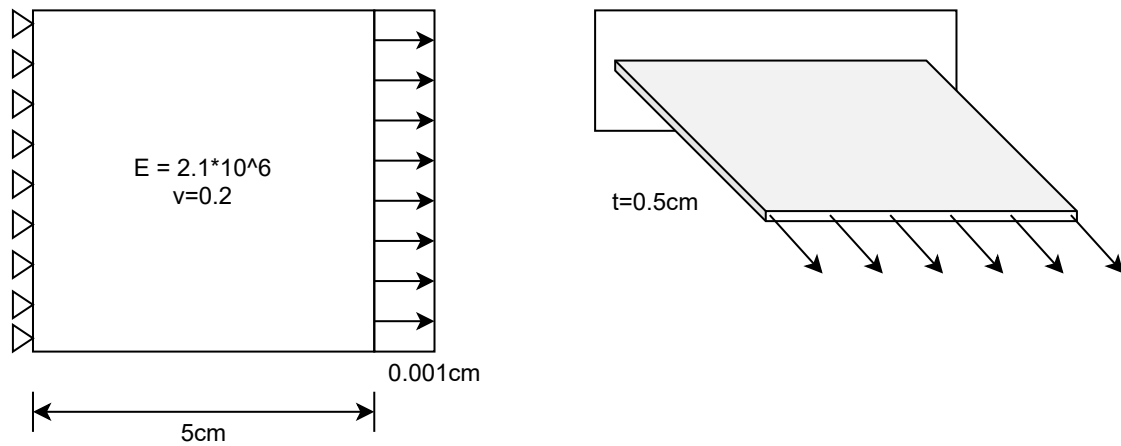


**Figura 8.** Diagrama de clases UML para el programa

## Casos de Estudio

Para poder evaluar el funcionamiento del programa y poder validar los resultados se parte de los casos de estudio propuestos por Pisano *et al.* (2009).

### 1. Placa a tensión:



**Figura 9.** Modelo de placa a tensión, tomado de Pisano *et al.* (2009)

Se tienen las siguientes características:

$$a = 5 \text{ cm}$$

$$t = 0.5 \text{ cm}$$

$$u_0 = 0.001 \text{ cm}$$

$$E = 2.1 * 10^6 \text{ daN/cm}^2$$

$$\nu = 0.2$$

Bajo las siguientes condiciones de borde:

$$U = 0, V = 0 \text{ en } x = 0$$

$$U = u_0 \text{ en } x = a$$

Usando el programa, este problema puede modelarse así:

```

1 import NLFEM
2 from NLFEM.Mesh import Rect
3 import matplotlib.pyplot as plt
4
5 RUTA_M = 'MATRICES'
6
7 E = 2.1*10**6
8 ν = 0.2
9 u = 0.001
10 a = 5 #Base del rectángulo
11 t = 0.5 #Grosor
12 l = 0.1
13
14 GEOMETRIA = Rect.Rect("enmallado.txt")
15 GEOMETRIA.generarSegmentosDesdeCoordenadas([0,0],[a,0])
16 GEOMETRIA.generarSegmentosDesdeCoordenadas([a,0],[a,a])
17 GEOMETRIA.generarSegmentosDesdeCoordenadas([a,a],[0,a])
18 GEOMETRIA.generarSegmentosDesdeCoordenadas([0,a],[0,0])
19 GEOMETRIA.dibujarse()

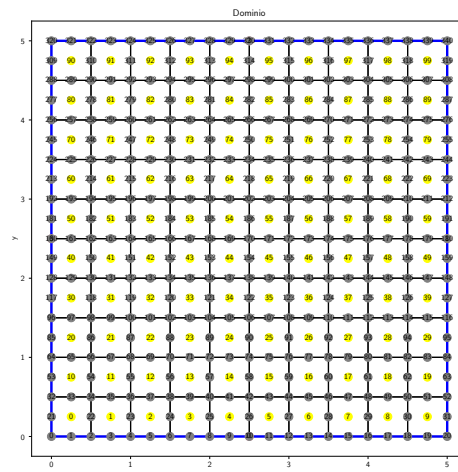
```

**Figura 10.** Importar librerías y enmallado



Se importa la librería NLFEM que contiene el programa de NL-FEM completo. Posteriormente, Se definen las variables iniciales del problema, por ejemplo, el módulo de Young, grosor, etc. Se carga la geometría del dominio mediante el archivo *enmallado.txt*, el cual sigue los lineamientos propuestos en la tabla 1. Por último, se definen los segmentos que serán de gran ayuda para imponer condiciones de borde. Cada segmento va de un punto inicial a un punto final. El programa encontrará el nodo del enmallado más cercano a las coordenadas pasadas por parámetro. Finalmente, en la línea 19 se grafica el enmallado final (la manera en cómo el programa está entendiendo el enmallado)

Para este caso, se trabaja con un enmallado de 100 elementos. El resultado obtenido se aprecia en la figura 11.



**Figura 11.** Enmallado de 100 elementos para dominio cuadrado

```

20 condiciones_borde_escenciales = GEOMETRIA.generarCBdesdeBorde(3,
  [0,0])+GEOMETRIA.generarCBdesdeBordeX(1,u)
21
22 Objeto_FEM = NLFEM.NoLocal(GEOMETRIA)
23 Objeto_FEM.z1 = 0.5
24 Objeto_FEM.moduloIntegrador("enmallado.txt",3,E,V,t,l,RUTA_M,tfa=1)
25 Objeto_FEM.generarElementos()
26 Objeto_FEM.importarMatricesCarpeta(RUTA_M)
27 Objeto_FEM.definirCondicionesDeBorde(condiciones_borde_escenciales)
28 Objeto_FEM.ensamblar()
29 Objeto_FEM.condicionesFrontera()
30 Objeto_FEM.solucionarSistemaEcuaciones()

```

**Figura 12.** Condiciones de borde integración y solución

En la línea 20 se nota el proceso de concatenado de las condiciones de borde, las cuales se generan mediante las funciones de la clase *Geometria*. Para este caso, se definen condiciones de borde en los segmentos 3 y 0. Los segmentos pueden observarse en la figura 11 con color azul. Los bordes se enumeran en el orden en que fueron agregados, lo cual se definió en las líneas 15-18 de la figura 10.

Se crea un objeto de la clase *NLFEM* que guarda toda la información necesaria para realizar el

proceso. Esta clase toma por parámetro la geometría. En la línea 23 se define el valor para  $\zeta_1$  que el programa usará para el ensamblaje de matrices.

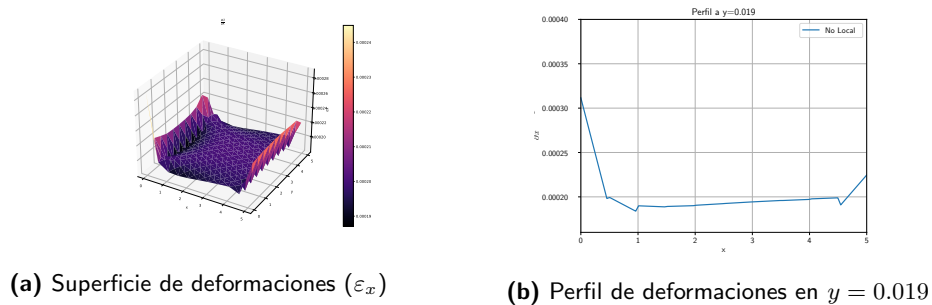
En la línea 24 se llama al módulo integrador. En la carpeta de la variable *RUTA\_M* se generarán los archivos de la integración que siguen el formato propuesto en la figura 7. Los parámetros del módulo integrador son los definidos previamente. Esta función se encarga de correr el archivo *index.exe* que contiene el código de C++ del módulo integrador. Cabe resaltar que este comportamiento solo será válido para Windows. Para otros sistemas operativos se deberá cambiar el archivo *index.exe* por el archivo compilado desde el código fuente desde C++.

En las líneas siguientes se ejecutan cada uno de los pasos para NLFEM mencionados anteriormente. Esto resuelve el problema de NL-FEM en el dominio.

```
31 import matplotlib.pyplot as plt
32 Objeto_FEM.defUnitariaX([10,10])
33 plt.show()
34 Objeto_FEM.perfily(0.019,0.00016,0.0004,0,a,acum=True,label='No
Local')
35 plt.show()
```

**Figura 13.** Post proceso

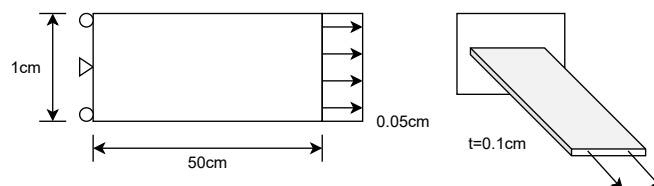
El programa permite realizar gráficas de resultados, en la figura 13 se muestran algunos de estas funciones. Las gráficas de resultados se muestran en la figura 14.



**Figura 14.** Resultados gráficos para un enmallado de 100 elementos

## 2. Barra a tensión:

La barra a tensión se puede modelar como se muestra en la figura 15.



**Figura 15.** Modelo de barra unidimensional tomado de Pisano *et al.* (2009)

Se tienen las siguientes características:

$$L = 50 \text{ cm}$$

$$h = 1 \text{ cm}$$

$$t = 0.1 \text{ cm}$$

$$u_0 = 0.05 \text{ cm}$$

$$E = 2.1 * 10^6 \text{ daN/cm}^2$$

$$\nu = 0.20$$

Bajo las siguientes condiciones de borde:

$$U = 0 \text{ en } x = 0$$

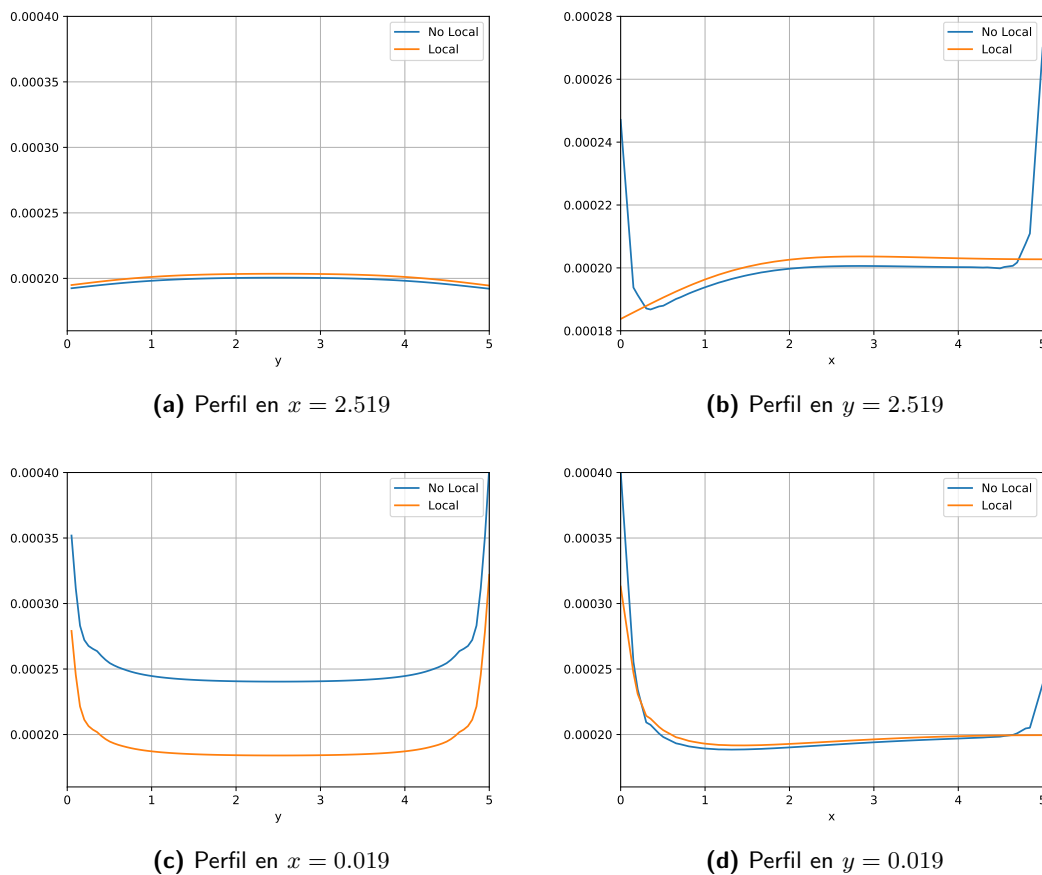
$$V = 0 \text{ en el punto } x = 0, y = h/2$$

$$U = u_0 \text{ en } x = L$$

Este problema puede ser modelado de la misma manera que el problema anterior, el cambio más relevante son las condiciones de borde (asumiendo los cambios obvios en propiedades del material y enmallado). Para ello basta con concatenar una condición de borde puntual con el comando `generarCBYdesdeCoordenada(x,y,valor)` que asignará la condición al nodo más cercano a la coordenada que se da por parámetro.

## Validación

Para realizar la validación de resultados obtenidos, se compararon los perfiles resultantes de correr el programa con los obtenidos por Pisano *et al.* (2009). En la figura 16 se muestran 4 perfiles obtenidos con el caso de estudio 1 usando una malla de 900 elementos,  $\zeta_1 = 0.5$ ,  $l = 0.1$ . Para comprobar la validez se realizó una superposición de imágenes entre los perfiles obtenidos y los perfiles propuestos por Pisano *et al.* (2009). Estas gráficas se encuentran en el anexo 18.



**Figura 16.** Perfiles de  $\varepsilon_x$  en el caso de estudio 1

## Variación de Parámetros

Para generar los resultados se itera con diferentes parámetros, el objetivo de cambiar los parámetros es ver la influencia que tienen en la solución.

### Variaciones Planteadas

#### 1. Variación en funciones de atenuación

El componente principal para evaluar serán las funciones de atenuación. Por tal razón, para cada función de atenuación se extraerán los resultados de forma gráfica. Se usarán las cuatro funciones de atenuación definidas en el capítulo 4.

#### 2. Variación en factor de rigidez ( $\zeta_1$ )

El factor de rigidez controla la proporción que tienen los efectos no locales. Dado que se puede expresar como porcentaje, se debe variar de 0 a 1.

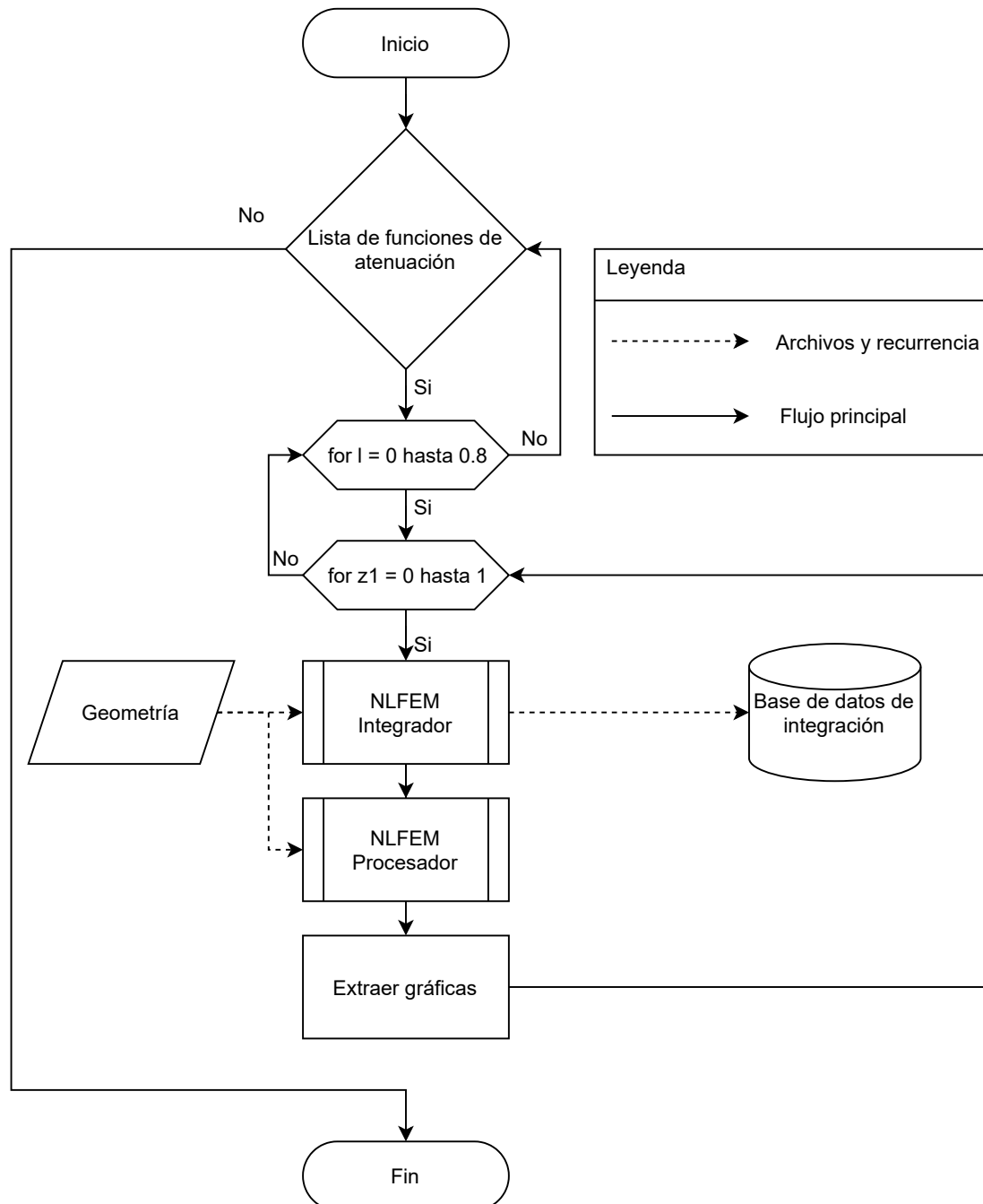
#### 3. Variación en longitud interna ( $l$ )

Dado que la longitud interna controla directamente la cantidad de elementos no locales, se estudiarán dos fenómenos. Cuando la cantidad de elementos no locales requeridas por el parámetro  $l$  es suficiente en el dominio y cuando la cantidad de elementos no locales requeridas por el parámetro  $l$  no es suficiente en el dominio. Por lo tanto, el parámetro  $l$  varía de 0 a la longitud

más pequeña del dominio sobre 12. En el caso del caso de estudio 1 se usará un  $l_{max} = \frac{1}{12}$  y en el caso de estudio 2  $l_{max} = \frac{1}{12}$ .

En el caso de estudio 1,  $l_{max}$  garantiza que para el elemento más interior hay suficientes elementos adyacentes en un radio  $Lr$  lo que garantiza que, aunque sea para ese elemento se tenga en cuenta todo el dominio. Los elementos que se encuentren fuera de este radio (las esquinas) no serán relevantes por estar demasiado lejos del elemento de interés (Polizzotto, 2001).

La metodología completa se resume en el diagrama de flujo presentado en la figura 17.

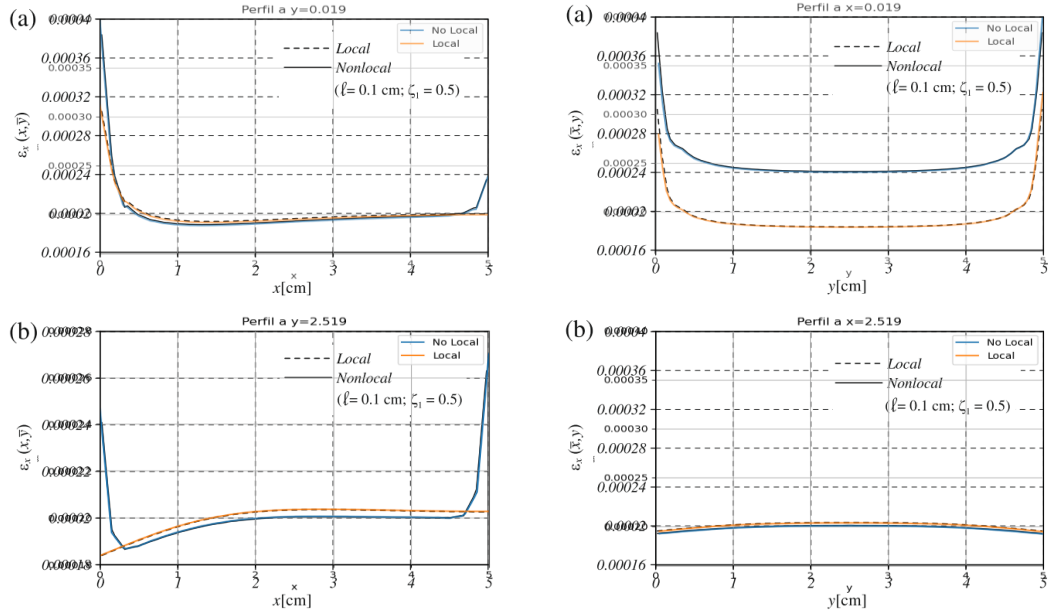


**Figura 17.** Diagrama de flujo del proceso iterativo

## Resultados

## Conclusiones

## Anexos



**Figura 18.** Gráficas de validación

$$UV_{nm}^{nloc}[i, j] = t^2 \int_{\Omega_n} \int_{\Omega_m} A(\rho) \left[ C_{12} \frac{\partial \psi_i^l}{\partial x} \frac{\partial \psi_j^{nl}}{\partial y} + C_{66} \frac{\partial \psi_i^l}{\partial y} \frac{\partial \psi_j^{nl}}{\partial x} \right] d\Omega_m d\Omega_n \quad (37a)$$

$$VU_{nm}^{nloc}[i, j] = t^2 \int_{\Omega_n} \int_{\Omega_m} A(\rho) \left[ C_{12} \frac{\partial \psi_i^l}{\partial y} \frac{\partial \psi_j^{nl}}{\partial x} + C_{66} \frac{\partial \psi_i^l}{\partial x} \frac{\partial \psi_j^{nl}}{\partial y} \right] d\Omega_m d\Omega_n \quad (37b)$$

$$VV_{nm}^{nloc}[i, j] = t^2 \int_{\Omega_n} \int_{\Omega_m} A(\rho) \left[ C_{11} \frac{\partial \psi_i^l}{\partial y} \frac{\partial \psi_j^{nl}}{\partial y} + C_{66} \frac{\partial \psi_i^l}{\partial x} \frac{\partial \psi_j^{nl}}{\partial x} \right] d\Omega_m d\Omega_n \quad (37c)$$

**Ecuación 37.** Modelo de elementos finitos segun la terminología de Reddy, 2005



## Referencias

- Altan, S. (1989). Uniqueness of initial-boundary value problems in nonlocal elasticity. *International Journal of Solids and Structures*, 25(11), 1271-1278. [https://doi.org/10.1016/0020-7683\(89\)90091-7](https://doi.org/10.1016/0020-7683(89)90091-7)
- Eringen, A. C. (1987). Theory of Nonlocal Elasticity and Some Applications. *Res mechanica*, 21(4), 313-342.
- Eringen, A. C. & Edelen, D. G. (1972). On nonlocal elasticity. *International Journal of Engineering Science*, 10(3), 233-248. [https://doi.org/10.1016/0020-7225\(72\)90039-0](https://doi.org/10.1016/0020-7225(72)90039-0)
- Kröner, E. (1967). Elasticity theory of materials with long range cohesive forces. *International Journal of Solids and Structures*, 3(5), 731-742. [https://doi.org/10.1016/0020-7683\(67\)90049-2](https://doi.org/10.1016/0020-7683(67)90049-2)
- Kuhlman, D. (2011). *A Python Book: Beginning Python, Advanced Python, and Python Exercises*. PLATYPUS GLOBAL MEDIA. <https://books.google.com.co/books?id=1FL-ygAACAAJ>
- Pisano, A., Sofi, A. & Fuschi, P. (2009). Nonlocal integral elasticity: 2D finite element based solutions. *International Journal of Solids and Structures*, 46(21), 3836-3849. <https://doi.org/10.1016/j.ijsolstr.2009.07.009>
- Polizzotto, C. (2001). Nonlocal elasticity and related variational principles. *International Journal of Solids and Structures*, 38(42-43), 7359-7380. [https://doi.org/10.1016/S0020-7683\(01\)00039-7](https://doi.org/10.1016/S0020-7683(01)00039-7)
- Ramirez, F. (2020). Meshing of a rectangular domain using 8-node elements identifying neighbor elements within a distance  $l_r$ .
- Reddy, J. N. D. (2005). *An Introduction to the Finite Element Method*. McGraw-Hill Education. <https://books.google.com.co/books?id=8gqnRwAACAAJ>
- Shewchuk, J. R. (1996). Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator [From the First ACM Workshop on Applied Computational Geometry]. En M. C. Lin & D. Manocha (Eds.), *Applied Computational Geometry: Towards Geometric Engineering* (pp. 203-222). Springer-Verlag.
- Zingales, M., Di Paola, M. & Inzerillo, G. (2011). The finite element method for the mechanically based model of non-local continuum. *International Journal for Numerical Methods in Engineering*, 86, 1558-1576. <https://doi.org/10.1002/nme.3118>