Nombre Fancy

David Arturo Rodriguez Herrera

Diciembre 2020

Índice

Índice de figuras

Índice de Tablas

Resumen

Introducción

Objetivos

Marco Teórico

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:

$$\sigma_{xz} = 0 \tag{1a}$$

$$\sigma_{yz} = 0 \tag{1b}$$

$$\sigma_{z} = 0 \tag{1c}$$

Lo cual permite expresar el tensor de rigidez C como:

$$\boldsymbol{C} = \begin{pmatrix} C_{11} & C_{12} & 0 \\ C_{12} & C_{11} & 0 \\ 0 & 0 & C_{66} \end{pmatrix} \tag{2}$$

Donde:

$$C_{11} = \frac{E}{1-v^2} \tag{3a}$$

$$C_{12} = \frac{vE}{1-v^2} \tag{3b}$$

$$C_{66} = \frac{E}{2(1+v)} \tag{3c}$$

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,comunmente llamado U y el segundo pertenece al desplazamiento en y, comunmente 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 mas 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 mas alejado del elemento. En la figura 1 se muestran gráficamente ambos procesos.

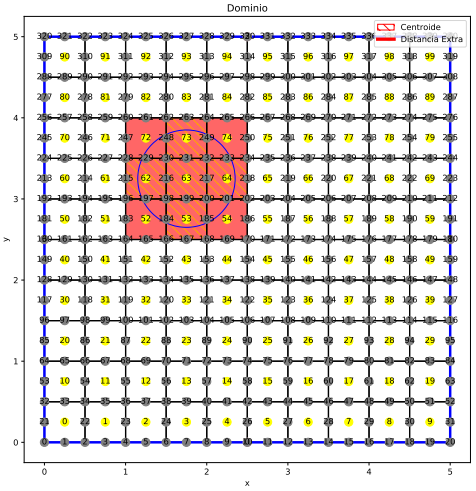


Figura 1. Cambios en la inclusión de elementos no locales. Se evalúa el elemento 63. El círculo azul representa un radio de $Lr = 6 * l$ con $l = 0.1$ desde el centroide del elemento

Dadas las propiedades de las funciones de atenuación descritas en el capítulo 4, se puede asegurar que si se toma una distancia $Lr \to \infty$ el resultado no cambia de manera significante. 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 $Lr \to \infty$ el costo computacional aumenta severamente, por lo que se aconseja siempre identificar los elementos no locales.

Para este estudio se tomara 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 numero 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 numero 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:

$$\boldsymbol{k_{nm}^{nloc}} = \int_{V_n}\int_{V_m} A(|\boldsymbol{x}-\boldsymbol{x'}|/l)\boldsymbol{B_n^T}(\boldsymbol{x})\boldsymbol{D}\boldsymbol{B_n}(\boldsymbol{x})dV'dV \tag{4}$$

   Donde:

   n se refiere al elemento local.

   m se refiere al elemento no local.

   A es la función de atenuación.

   $\boldsymbol{B_n}$ es la matriz de derivadas parciales de las funciones de forma.

   $\boldsymbol{D}$ tensor de rigidez.

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

   $\boldsymbol{x'}$ 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 $\boldsymbol{K_{uu}}$:

$$\boldsymbol{UU_{nm}^{nloc}}[i,j] = t^2\int_{\Omega_n}\int_{\Omega_m} A(\rho)\left[C_{11}\frac{\partial\psi_i^l}{\partial x}\frac{\partial\psi_j^{nl}}{\partial x} + C_{66}\frac{\partial\psi_i^l}{\partial y}\frac{\partial\psi_j^{nl}}{\partial y}\right]d\Omega_m d\Omega_n \tag{5}$$

   Notese que a la ecuación 4 es una integral volumétrica, mientras que la ecuación 5 corresponde a una integral de área, por tal razon, se debe multiplicar por el grosor al cuadrado.

   Donde:

   ρ es la relación $|\boldsymbol{x}-\boldsymbol{x'}|/l$.

   ψ_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 estan evaluadas en los puntos correspondientes a su elemento, es decir, la función ψ_i^{nl} esta evaluada en los puntos x_{nl}, y_{nl}

   En los anexos 6 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 ensablaje 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 el 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 esta escrito en Python y requiere de las librerías *triangle*, *NumPy* y *Matplotilib*. 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 5 y sus variaciones en el anexo 6.

   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 reconzca 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 5. 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 libreria 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 mas costoso computacional mente es la integración de las matrices no locales, por tal razon, se decidió realizar un segundo módulo que se encargue unicamente 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 (*int*) siguiendo los lineamientos del capitulo 4

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 modulo 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 computacional mente 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 2

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}

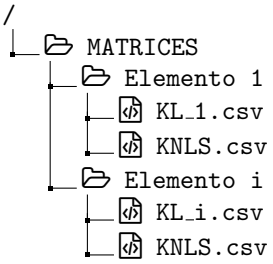


Figura 2. Estructura de guardado de archivos para las matrices

4. Estructura final

   El programa completo se orientó a objetos y su estructura se muestra en el diagrama UML de la figura 3.

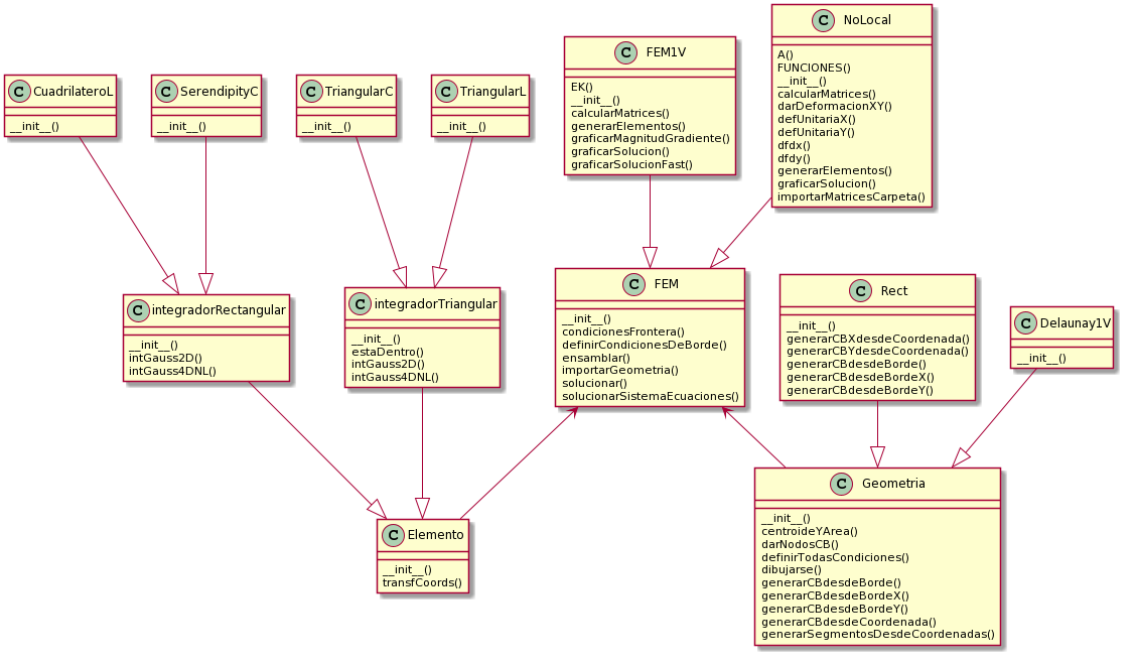


Figura 3. Diagrama de clases UML para el programa

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:**

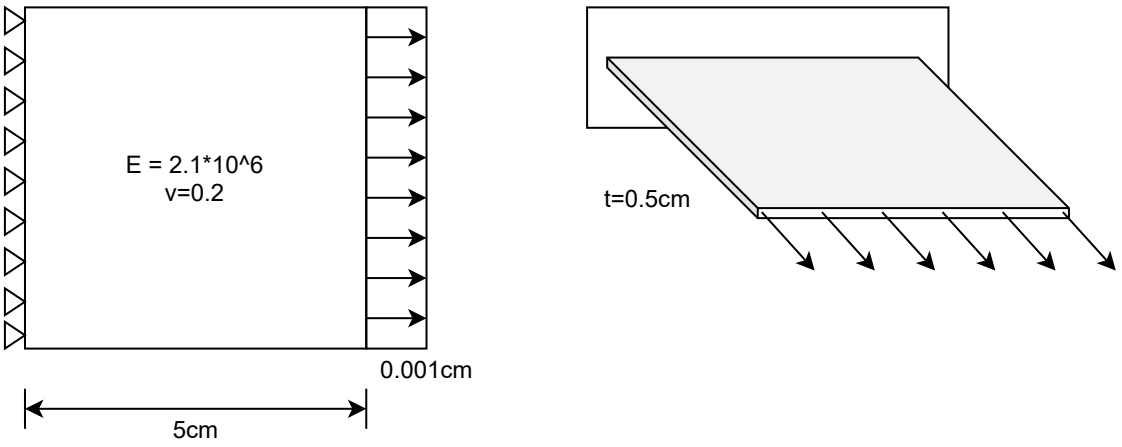


Figura 4. Modelo de placa a tensión, tomado de Pisano *et al.* 2009

Se tienen las siguientes características:

$a = 5\ cm$

$t = 0.5\ cm$

$u_0 = 0.001\ cm$

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

$\nu = 0.2$

Bajo las siguientes condiciones de borde:

$U = 0, V = 0$ en $x = 0$

$U = u_0$ en $x = a$

Usando el programa, este problema puede modelarse así:

```python
import NLFEM
from NLFEM.Mesh import Rect
import matplotlib.pyplot as plt

RUTA_M = 'MATRICES'

E = 2.1*10**6
V = 0.2
u = 0.001
a = 5 #Base del rectángulo
t = 0.5 #Grosor
l = 0.1

GEOMETRIA = Rect.Rect("enmallado.txt")
GEOMETRIA.generarSegmentosDesdeCoordenadas([0,0],[a,0])
GEOMETRIA.generarSegmentosDesdeCoordenadas([a,0],[a,a])
GEOMETRIA.generarSegmentosDesdeCoordenadas([a,a],[0,a])
GEOMETRIA.generarSegmentosDesdeCoordenadas([0,a],[0,0])
GEOMETRIA.dibujarse()
```

Figura 5. 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 mas cercano a las coordenadas pasadas por parámetro. Finalmente, en la línea 19 se grafíca el enmallado final (la manera en como el programa esta entendiendo el enmallado)

Para este caso, se trabaja con un enmallado de 100 elementos. El resultado obtenido se aprecia en la figura 6

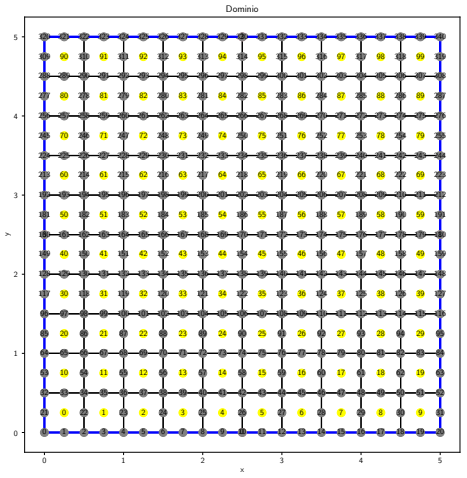


Figura 6. Enmallado de 100 elementos para dominio cuadrado.

```
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 7. 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 6 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 5.

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 ζ_1 que el programa usará para el ensamblaje de matrices.

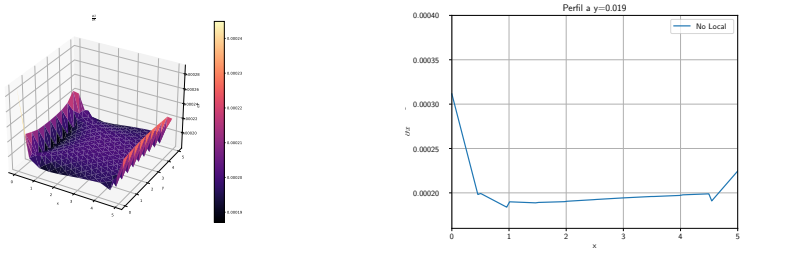
En la línea 24 se llama al módulo integrador. En la carpeta de la variable *RUTA_M* se generarán los archivos de la integración que siguen el formato propuesto en la figura 2. Los parámetros del módulo integrador son los definidos previamente. Esta función se encarga de correr el archivo *index.exe* que contiene el codigo 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 8. Post proceso

El programa permite realizar gráficas de resultados, en la figura 8 se muestran algunos de estas funciones. Las gráficas de resultados se muestran en la figura 9.



(a) Superficie de deformaciones (ε_x)

(b) Perfil de deformaciones en $y = 0.019$

Figura 9. 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 10

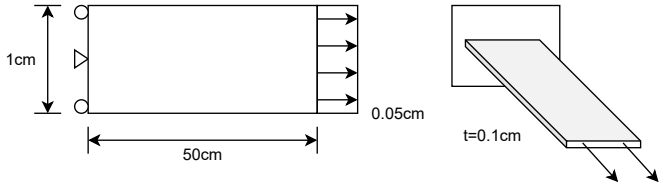


Figura 10. Modelo de barra unidimensional tomado de Pisano *et al.* 2009

Se tienen las siguientes características:

$L = 50\ cm$

$h = 1\ cm$

$t = 0.1\ cm$

$u_0 = 0.05\ cm$

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

$\nu = 0.20$

Bajo las siguientes condiciones de borde:

$U = 0$ en $x = 0$

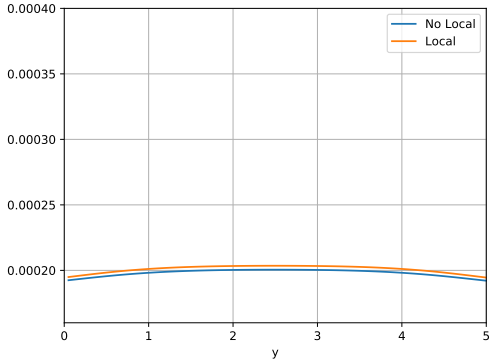
$V = 0$ en el punto $x = 0, y = h/2$

$U = u_0$ en $x = L$

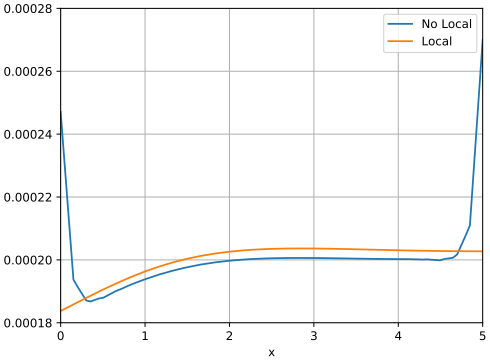
Este problema puede ser modelado de la misma manera que el problema anterior, el cambio mas relevante son las condiciones de borde (asumiendo los cambios obvios en propiedades del material y enmallado). Para ello basca con concatenar una condicion de borde puntual con el comando *generarCBYdesdeCoordenada(x,y,valor)*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

Validación

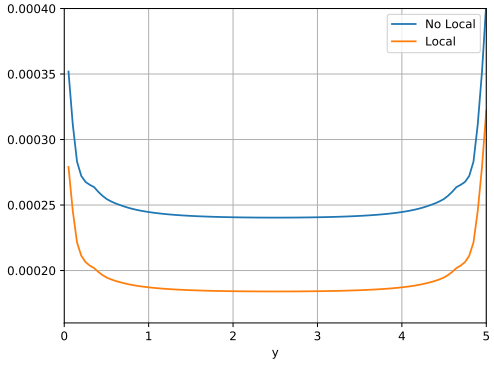
Para realizar la validación de resultados obtenidos, se compararon los perfiles resultantes de correr el programa con los obtenidos por Pisano *et al.* 2009. En la figura 11 se muestran 4 perfiles obtenidos con el caso de estudio 1 usando una malla de 900 elementos, $\zeta_1 = 0.5$, $l = 0.1$. Para comprobar la validez se realizo una superposición de imágenes entre los perfiles obtenidos y los perfiles propuestos por Pisano *et al.* 2009. Estas gráficas se encuentran en el anexo 13



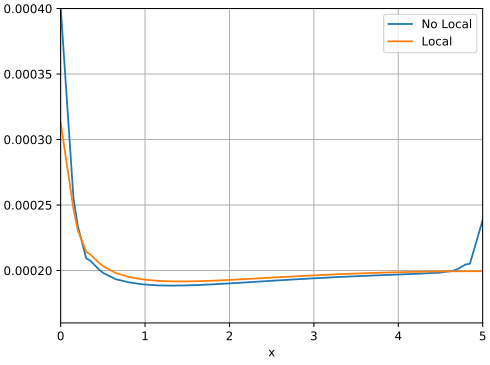
(a) Perfil en $x = 2.519$



(b) Perfil en $y = 2.519$



(c) Perfil en $x = 0.019$



(d) Perfil en $y = 0.019$

Figura 11. Perfiles de ε_x en el caso de estudio 1

Variación de Parámetros

Para generar los resultados se itera condiferentes 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 a evaluar serán las funciones de atenuación. Por tal razon, 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 (ζ_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 parametro l varía de 0 a la lonigitud

mas pequeña del dominio sobre 12. En el caso del caso de estudio 1 se usara 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 mas interior hay suficientes elementos adyacentes en un radio Lr lo que garantiza que aunquesea 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 12

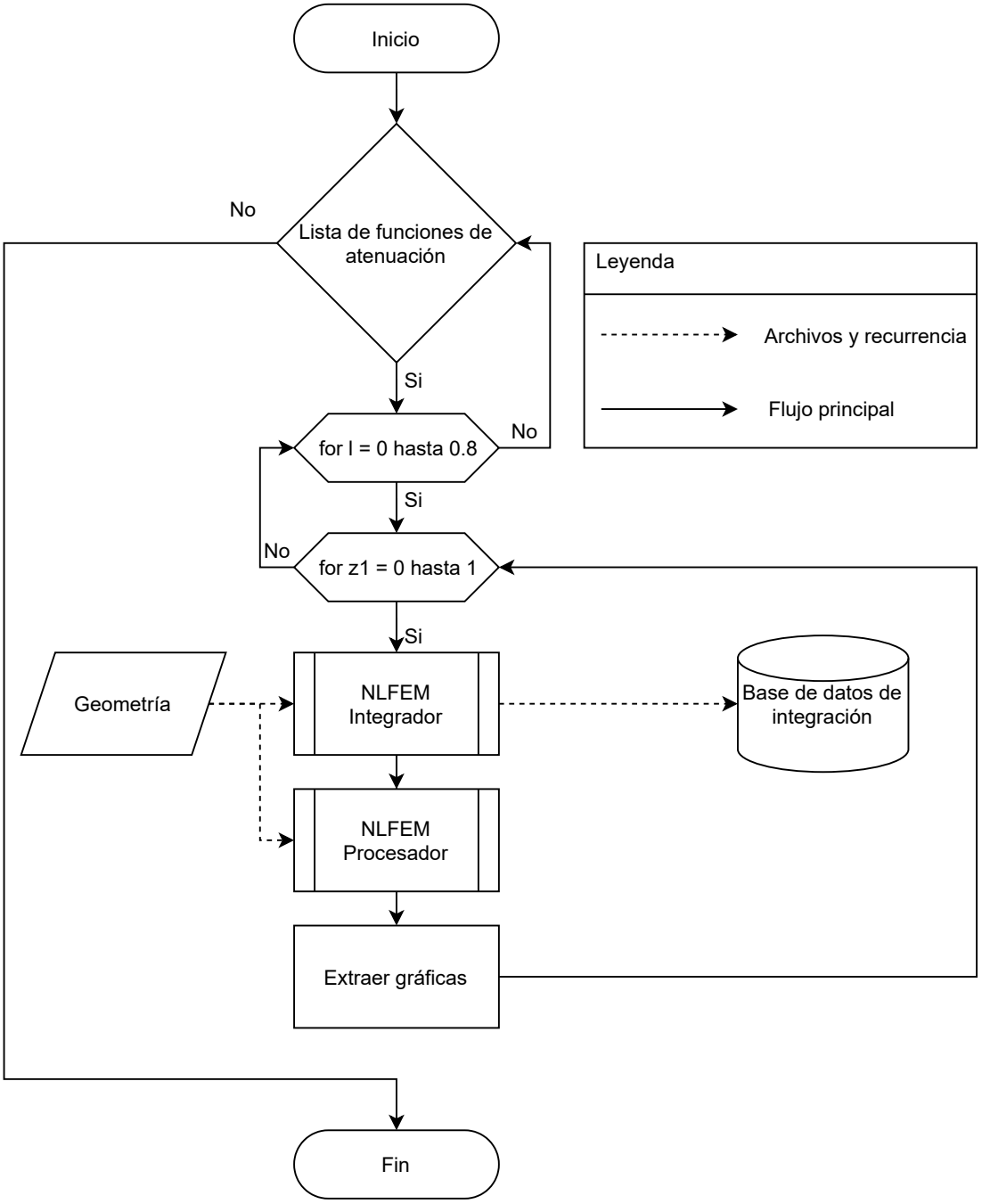


Figura 12. Diagrama de flujo del proceso iterativo

Resultados

Conclusiones

Anexos

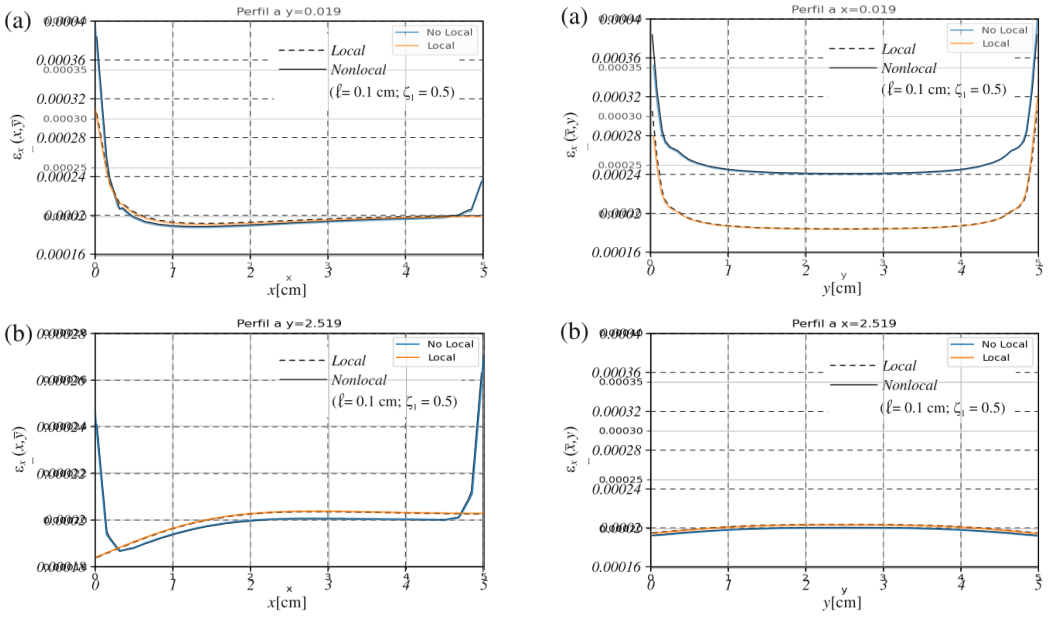


Figura 13. Gráficas de validación

$$\boldsymbol{UV}^{\boldsymbol{nloc}}_{\boldsymbol{nm}}[i,j] = t^2 \int_{\Omega_n}\int_{\Omega_m} A(\rho)\left[C_{12}\frac{\partial\psi_i^l}{\partial x}\frac{\partial\psi_j^{nl}}{\partial y} + C_{66}\frac{\partial\psi_i^l}{\partial y}\frac{\partial\psi_j^{nl}}{\partial x}\right]d\Omega_m d\Omega_n \tag{6a}$$

$$\boldsymbol{VU}^{\boldsymbol{nloc}}_{\boldsymbol{nm}}[i,j] = t^2 \int_{\Omega_n}\int_{\Omega_m} A(\rho)\left[C_{12}\frac{\partial\psi_i^l}{\partial y}\frac{\partial\psi_j^{nl}}{\partial x} + C_{66}\frac{\partial\psi_i^l}{\partial x}\frac{\partial\psi_j^{nl}}{\partial y}\right]d\Omega_m d\Omega_n \tag{6b}$$

$$\boldsymbol{VV}^{\boldsymbol{nloc}}_{\boldsymbol{nm}}[i,j] = t^2 \int_{\Omega_n}\int_{\Omega_m} A(\rho)\left[C_{11}\frac{\partial\psi_i^l}{\partial y}\frac{\partial\psi_j^{nl}}{\partial y} + C_{66}\frac{\partial\psi_i^l}{\partial x}\frac{\partial\psi_j^{nl}}{\partial x}\right]d\Omega_m d\Omega_n \tag{6c}$$

Ecuación 6. Modelo de elementos finitos segun la terminología de Reddy, 2005

Referencias

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

Ramirez, F. (2020). Meshing of a rectangular domain using 8-node elements identifying neighbor elements within a distance lr.

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