Politecnico di Torino

Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Meccanica
A.a. 2023/2024
Sessione di Laurea Aprile 2024

Tesi di Laurea Magistrale

Risoluzione di problemi di deformazione elastica accoppiati 2D-1D su mesh non conformi con un metodo basato sull'ottimizzazione vincolata da E.D.P (Equazioni alle derivate parziali)

Relatore:
Stefano Scialò

Candidato:
Giuseppe Fornelli

INDICE

INDICE FIGURE

ABSTRACT

Si propone un approccio numerico per la simulazione di equazioni ellittiche accoppiate bidimensionali e unidimensionali (accoppiamento 2D-1D) derivanti dalla riduzione della dimensione geometrica di problemi 2D-2D con inclusioni sottili, ovvero con una larghezza molto inferiore alla loro lunghezza. Questo permette di ridurre il costo computazionale legato alla risoluzione del problema, evitando di generare un mesh all'interno delle inclusioni. Allo stesso tempo il metodo prevede l'uso di mesh non conformi per il dominio 2D e le inclusioni 1D, ovvero le inclusioni possono attraversare in modo arbitrario gli elementi della mesh bidimensionale e non è richiesta alcuna corrispondenza tra gli elementi della mesh 2D e 1D, aumentando così la flessibilità del metodo. Tutto ciò è ottenuto mediante la minimizzazione di una funzione appositamente progettata per imporre condizioni di accoppiamento alle interfacce tra i problemi 2D e 1D. Il metodo è implementato in linguaggio Matlab facendo uso del paradigma della programmazione ad oggetti e viene applicato, in particolare, allo studio delle equazioni di deformazione elastica, analizzando alcuni esempi specifici.

INTRODUZIONE

Lo studio in questione applica un approccio numerico per gestire l'accoppiamento tra equazioni ellittiche bidimensionali e unidimensionali (accoppiamento 2D-1D), facendo riferimento, nello specifico, a problemi di deformazione elastica. Tale necessità emerge, ad esempio, nel trattamento numerico di domini contenenti piccole inclusioni: in questi casi, potrebbe essere conveniente approssimare le piccole inclusioni con entità unidimensionali (1D) per evitare la creazione di una griglia bidimensionale all'interno dell'inclusione. Questa riduzione è fattibile solo se le assunzioni di modellazione unidimensionale possono essere applicate al problema specifico. Esempio di applicazione è la modellazione di materiali rinforzati con fibre.

Nel lavoro che segue si propone un approccio numerico con il quale si riesce ad ottenere un modello ridotto 2D-1D che approssima il problema 2D-2D, introducendo ipotesi appropriate sulla soluzione all'interno delle piccole inclusioni.

I problemi nel dominio 2D e nelle piccole inclusioni sono separati utilizzando un metodo di decomposizione del dominio basato su tre campi, con condizioni di interfaccia appropriate per garantire la continuità della soluzione. Questo approccio offre vantaggi rispetto ad altri metodi di decomposizione del dominio, come la definizione di schemi numerici localmente conservativi su mesh non conformi e la computazione diretta delle variabili di interfaccia. Le condizioni di interfaccia sono applicate tramite un metodo di ottimizzazione numerica vincolata da PDE, che permette di imporre il vincolo di continuità usando mesh completamente indipendenti nei sottodomini, offrendo flessibilità e robustezza anche in presenza di complessità geometriche.

Nella trattazione vengono innanzitutto introdotti cenni sulla deformazione elastica sia per il caso del filo elastico che per il caso della membrana elastica, ricavandone le

equazioni costitutive che saranno alla base dello studio. Si procede, poi, con l'approfondimento del metodo degli elementi finiti per entrambi i casi ricavando a partire dalla formulazione variazionale del problema continuo, la formulazione del problema discreto. È possibile, così, con questi elementi di base trattare il problema vero e proprio riguardante i sistemi accoppiati 2D-1D. Si riduce un modello equidimensionale ad un modello bi-dimensionale, potendo utilizzare mesh non conformi e riducendo così notevolmente il costo computazionale. Ciò è possibile grazie alla minimizzazione di un funzionale creato appositamente per lo studio di accoppiamento alle interfacce di problemi 2D-1D. Infine, dopo una breve introduzione al paradigma della programmazione ad oggetti, utilizzata nella programmazione in ambiente MATLAB, vengono mostrate le parti caratterizzati dei codici sviluppati e analizzati i risultati numerici in differenti casistiche, dimostrando la validità del metodo studiato.

CENNI SULLA DEFORMAZIONE ELASTICA NEL CASO DELLA MEMBRANA E DEL FILO ELASTICO

Analizziamo le equazioni di deformazione elastica in due contesti specifici: un filo elastico e una membrana elastica. Questi due sistemi rappresentano esempi di problemi elastici in una dimensione (1D) e in due dimensioni (2D), rispettivamente.

Filo Elastico (1D)

Sia dato un filo elastico sottile di sezione circolare costante S. In assenza di forze esterne, il suo asse occupa la posizione dell'intervallo [0, L] lungo l'asse delle coordinate x_1. L'elastico è fissato alle estremità dell'intervallo.

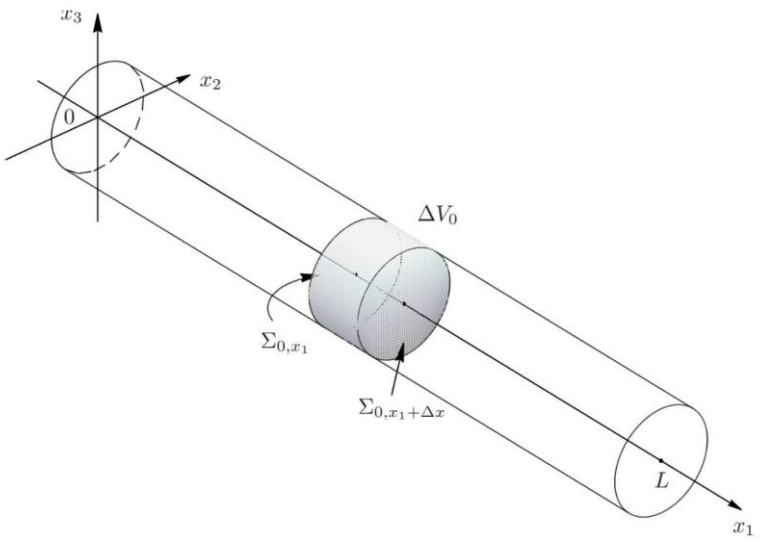


Figura 1 Filo elastico

Applichiamo al filo una (piccola) densità volumica di forza $f = 0e_1 + 0e_2 + f_3e_3$, giacente nel piano x_1x_3 e diretta normalmente all'asse del filo. Essa induce un (piccolo) spostamento $u = u_2e_1 + u_2e_2 + u_3e_3$ del filo a partire dalla posizione di riferimento; precisamente, $u = u(x)$ denota lo spostamento della particella materiale che nella posizione di riferimento si trova nel punto x.

Lo spostamento sarà complanare con la forza, dunque $u_2 = 0$; inoltre, in prima approssimazione la componente u_1 sarà trascurabile rispetto alla componente u_3 che descrive lo spostamento nella direzione della forza.

Sia $x_1 \in (0, L)$ e sia $\Delta V_0 = [x_1, x_1 + \Delta x] \times S$ un elemento di filo elastico di lunghezza Δx, nella posizione di riferimento; siano $\Sigma_{0,x1} = \{x_1\} \times S$ e $\Sigma_{0,x1+\Delta x} = \{x_1 + \Delta x\} \times S$ le sezioni che delimitano l'elemento.

Esso si trasforma nell'elemento ΔV per effetto della forza applicata; siano Σ_{x1} e $\Sigma_{x1+\Delta x}$ le sezioni trasformate. Indichiamo con

$$\underline{\boldsymbol{\sigma}} = \begin{pmatrix} \sigma_1 & \tau_{12} & \tau_{13} \\ \tau_{21} & \sigma_2 & \tau_{23} \\ \tau_{31} & \tau_{23} & \sigma_3 \end{pmatrix}$$

il tensore degli sforzi del filo. Sia poi n la normale alla sezione trasformata del filo, orientata nel senso crescente delle x_1. L'equazione di equilibrio è

$$\int_{\Delta V} \boldsymbol{f}\, dV + \int_{\Sigma_{x_1+\Delta x}} \underline{\boldsymbol{\sigma}}\, \boldsymbol{n}\, d\Sigma - \int_{\Sigma_{x_1}} \underline{\boldsymbol{\sigma}}\, \boldsymbol{n}\, d\Sigma = \mathbf{0}\ .$$

Nell'ipotesi di spostamenti piccoli, possiamo in prima approssimazione confondere ΔV con ΔV_0 e le sezioni trasformate con quelle di riferimento. Questo e l'ipotesi di filo sottile ci permette di passare a un modello monodimensionale. Infatti

$$\int_{\Delta V} \boldsymbol{f}\, dV \simeq \int_{\Delta V_0} \boldsymbol{f}\, dV = \int_{x_1}^{x_1+\Delta x} \left(\int_S \boldsymbol{f}(x, x_2, x_3)\, dx_2 dx_3 \right) dx = |S| \int_{x_1}^{x_1+\Delta x} \tilde{\boldsymbol{f}}(x)\, dx\ ,$$

avendo indicato con |S| l'area della sezione S e con $\tilde{f}(x)$ il valor medio di f sulla sezione S nel punto $x \in (0, L)$:

$$\tilde{\boldsymbol{f}}(x_1) = \frac{1}{|S|}\int_S \boldsymbol{f}(x_1, x_2, x_3)\, dx_2 dx_3 \ .$$

Procedendo in modo simile con gli integrali di superficie arriviamo alle equazioni approssimate

$$|S|\int_{x_1}^{x_1+\Delta x} \tilde{\boldsymbol{f}}(x)\, dx + |S|\,\underline{\tilde{\boldsymbol{\sigma}}}\,\tilde{\boldsymbol{n}}_{|x_1+\Delta x} - |S|\,\underline{\tilde{\boldsymbol{\sigma}}}\,\tilde{\boldsymbol{n}}_{|x_1} = \boldsymbol{0}\ .$$

Dividiamo per |S| e, per non appesantire le notazioni, cancelliamo il simbolo ˜; inoltre notiamo che, per le ipotesi fatte, $n \approx e_1$. Otteniamo quindi

$$\int_{x_1}^{x_1+\Delta x} \boldsymbol{f}(x)\, dx + \underline{\boldsymbol{\sigma}}\,\boldsymbol{e}_{1\,|x_1+\Delta x} - \underline{\boldsymbol{\sigma}}\,\boldsymbol{e}_{1\,|x_1} = \boldsymbol{0}\ .$$

Prendiamo ora la componente di tale equazione lungo l'asse x_3, ossia formiamone il prodotto scalare con e_3, ottenendo

$$\int_{x_1}^{x_1+\Delta x} f_3(x)\, dx + \tau_{31\,|x_1+\Delta x} - \tau_{31\,|x_1} = 0\ .$$

Dividendo per Δx e passando al limite per Δx → 0, arriviamo alla relazione differenziale

$$f_3(x_1) + \frac{d\tau_{31}}{dx}(x_1) = 0\ , \qquad x_1 \in (0, L)\ ,$$

che esprime la condizione di equilibrio del filo.

Invochiamo ora la legge costitutiva di Hooke che lega il tensore degli sforzi σ al tensore delle deformazioni ε:

$$\underline{\boldsymbol{\sigma}} = 2\mu\underline{\boldsymbol{\varepsilon}} + \lambda \mathrm{tr}(\underline{\boldsymbol{\varepsilon}})\underline{\boldsymbol{I}} \ ,$$

dove $\lambda > 0$, $\mu > 0$ sono i coefficienti di Lamé del materiale elastico, ε è il tensore delle deformazioni

$$\underline{\boldsymbol{\varepsilon}} = (\varepsilon_{ij})_{1\leq i,j\leq 3} \ , \qquad \varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \ ,$$

$\mathrm{tr}(\varepsilon) = \varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33} = \nabla \cdot u$ è la traccia del tensore delle deformazioni e $I = (\delta_{ij})_{1\leq i,j\leq 3}$ è il tensore identità. Prendendo la componente 3,1 dell'equazione costitutiva di Hooke e ricordando che u_1 è trascurabile rispetto a u_3, otteniamo l'equazione costitutiva approssimata

$$\tau_{31} = \mu\frac{\partial u_3}{\partial x_1} \ .$$

Infine, per semplicità omettiamo i pedici alle quantità u_3, τ_{31} e f_3 introdotte finora. Il coefficiente μ, detto modulo di taglio, può essere espresso in funzione del modulo di Young E e del coefficiente di Poisson ν come

$$\mu = \frac{E}{2(1+\nu)} \ .$$

Siamo dunque arrivati al seguente sistema di equazioni:

$$\begin{cases} \dfrac{d\tau}{dx} + f = 0 & \text{in } (0, L) \ , \\ \tau = \mu \dfrac{du}{dx} & \text{in } (0, L) \ , \\ u(0) = u(L) = 0 \ , \end{cases}$$

ove l'ultima relazione esprime il fatto che il filo è fissato alle estremità dell'intervallo [0, L]. Sostituendo l'espressione di τ nella prima equazione otteniamo

$$\begin{cases} -\dfrac{d}{dx}\left(\mu \dfrac{du}{dx}\right) = f & \text{in } (0, L) \ , \\ u(0) = u(L) = 0 \ . \end{cases}$$

Lo spostamento del filo elastico è dunque soluzione di un problema ai valori al bordo (o al contorno) per una equazione differenziale lineare del secondo ordine. Tale problema ammette una e una sola soluzione, se ad esempio μ ed f sono funzioni continue (o continue a tratti) in [0, L].

Membrana Elastica (2D)

Una membrana elastica è un sistema bidimensionale che può subire deformazioni sotto l'azione di forze distribuite o localizzate.

Il modello della membrana elastica estende alle due dimensioni spaziali quello del filo elastico; la sua deduzione è analoga. Consideriamo una membrana elastica sottile, la cui sezione mediana occupi in assenza di forze esterne una regione limitata Ω del piano $x_1 x_2$.

Supponiamo che la membrana sia fissata lungo tutto il suo bordo ∂Ω.

Una (piccola) densità volumica di forza $f = 0e_1 + 0e_2 + f_3e_3$, diretta normalmente alla sezione mediana della membrana induce un (piccolo) spostamento $u = u_2e_1 + u_2e_2 + u_3e_3$ a partire dalla posizione di riferimento.

In prima approssimazione le componenti u_1 e u_2 saranno trascurabili rispetto alla componente u_3 che descrive lo spostamento nella direzione della forza.

Poniamo $f = f_3$, $u = u_3$ e sia $\tau = (\tau_{31}, \tau_{32})$ il vettore delle componenti verticali dello sforzo di taglio, e $\mu > 0$ il modulo di taglio del materiale.

La condizione di equilibrio della membrana si esprime come

$$f + \left(\frac{\partial \tau_{31}}{\partial x} + \frac{\partial \tau_{32}}{\partial y} \right) = 0 \qquad \text{ossia} \qquad f + \nabla \cdot \boldsymbol{\tau} = 0 \, ,$$

mentre le equazioni costitutive (approssimate) sono date da

$$\tau_{31} = \mu \frac{\partial u}{\partial x} \, , \quad \tau_{32} = \mu \frac{\partial u}{\partial y} \, , \qquad \text{ossia} \qquad \boldsymbol{\tau} = \mu \nabla u \, .$$

Esse valgono all'interno di Ω, mentre sul bordo $\partial\Omega$ vale la condizione $u = 0$, che esprime il fatto che la membrana è fissata lungo tutto il bordo.

Otteniamo quindi il sistema

$$\begin{cases} \nabla \cdot \boldsymbol{\tau} + f = 0 & \text{in } \Omega \, , \\ \boldsymbol{\tau} = \mu \nabla u & \text{in } \Omega \, , \\ u = 0 & \text{su } \partial\Omega \, . \end{cases}$$

Sostituendo nella prima equazione l'espressione di τ data dalla seconda equazione, otteniamo il problema ai valori al bordo di Dirichlet

$$\begin{cases} -\nabla \cdot (\mu \nabla u) = f & \text{in } \Omega\,, \\ u = 0 & \text{su } \partial\Omega\,. \end{cases}$$

nell'incognita u. Notiamo che l'equazione in Ω si esplicita come

$$-\frac{\partial}{\partial x}\left(\mu \frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(\mu \frac{\partial u}{\partial y}\right) = f\,.$$

Nel caso in cui il coefficiente μ sia costante in Ω, otteniamo l'equazione di Poisson

$$-\mu\,\Delta u = f \qquad \text{in } \Omega\,,$$

dove l'espressione

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

indica il Laplaciano della funzione u.

I modelli di filo elastico e membrana elastica illustrano come le equazioni di deformazione elastica possano essere applicate a sistemi di diversa dimensionalità. Mentre un filo elastico offre un modello semplificato per studiare le deformazioni unidimensionali, la membrana elastica permette di esplorare le dinamiche di deformazione in due dimensioni. Entrambi i sistemi sono fondamentali per comprendere principi più complessi nell'elasticità e nella meccanica dei solidi.

FONDAMENTI DEL METODO DEGLI ELEMENTI FINITI

Il metodo degli elementi finiti è una tecnica numerica per trovare approssimazioni alle soluzioni di problemi di valore al contorno.

Nel caso di equazioni di deformazione elastica lo studio si presenta nella metodologia seguente.

Supponiamo che il bordo $\partial\Omega$ sia diviso in una parte non vuota Γ_D, su cui imponiamo una condizione di Dirichlet non omogenea, e nel complementare Γ_N, su cui imponiamo una condizione di Neumann non omogenea. Il problema è quindi

$$\begin{cases} -\nabla \cdot (\mu \nabla u) = f & \text{in } \Omega \,, \\ u = g & \text{su } \Gamma_D \,, \\ \mu \dfrac{\partial u}{\partial n} = \psi & \text{su } \Gamma_N \,, \end{cases}$$

con g e ψ funzioni assegnate.

La condizione di Neumann rappresenta l'assegnazione su Γ_N della componente normale dello sforzo di taglio. L'insieme V (g) degli spostamenti ammissibili, o funzioni di forma, è ora composto da funzioni che valgono g su Γ_D, mentre assumono valori arbitrari su Γ_N.

L'insieme V (0) delle funzioni nulle su Γ_D sarà l'insieme delle variazioni ammissibili, o funzioni test.

La formulazione variazionale del problema con le nuove condizioni al bordo si basa sul teorema della divergenza

$$-\int_{\Omega} \nabla \cdot (\mu \nabla u)\, v\, d\boldsymbol{x} = \int_{\Omega} \mu \nabla u \cdot \nabla v\, d\boldsymbol{x} - \int_{\partial\Omega} \mu \frac{\partial u}{\partial n}\, v\, ds\ ,$$

in cui u ∈ V (g) è la soluzione del nostro problema mentre $v \in$ V (0) è una qualunque funzione test. L'integrale sul bordo a secondo membro può essere scritto come

$$\int_{\partial\Omega} \mu \frac{\partial u}{\partial n}\, v\, ds = \int_{\Gamma_D} \mu \frac{\partial u}{\partial n}\, v\, ds + \int_{\Gamma_N} \mu \frac{\partial u}{\partial n}\, v\, ds = 0 + \int_{\Gamma_N} \psi\, v\, ds\ ,$$

avendo tenuto conto che v si annulla su Γ_D e che u soddisfa la condizione di Neumann su Γ_N . Pertanto, la formulazione variazionale del problema è

$$\begin{cases} u \in V(g) \text{ e soddisfa} \\ \displaystyle\int_{\Omega} \mu \nabla u \cdot \nabla v\, d\boldsymbol{x} = \int_{\Omega} f v\, d\boldsymbol{x} + \int_{\Gamma_N} \psi\, v\, ds \qquad \text{per ogni } \ v \in V(0)\ . \end{cases}$$

Dividiamo, dunque, una regione dello spazio in un numero finito di pezzi più piccoli, noti come elementi finiti (spesso triangoli) i quali, attraverso la loro aggregazione, formano una rappresentazione semplificata del dominio di studio.

Supponiamo d'ora in avanti che $\bar{\Omega} = \Omega + \partial\Omega$ sia un poligono. Decomponiamo $\bar{\Omega}$ nell'unione di un numero finito di triangoli non degeneri T(gli elementi geometrici del metodo), soddisfacenti la seguente condizione di ammissibilità:

- l'intersezione di due triangoli distinti può soltanto essere un intero lato comune ad entrambi i triangoli, oppure
- un vertice comune ad entrambi i triangoli, oppure
- l'insieme vuoto.

Situazioni proibite:

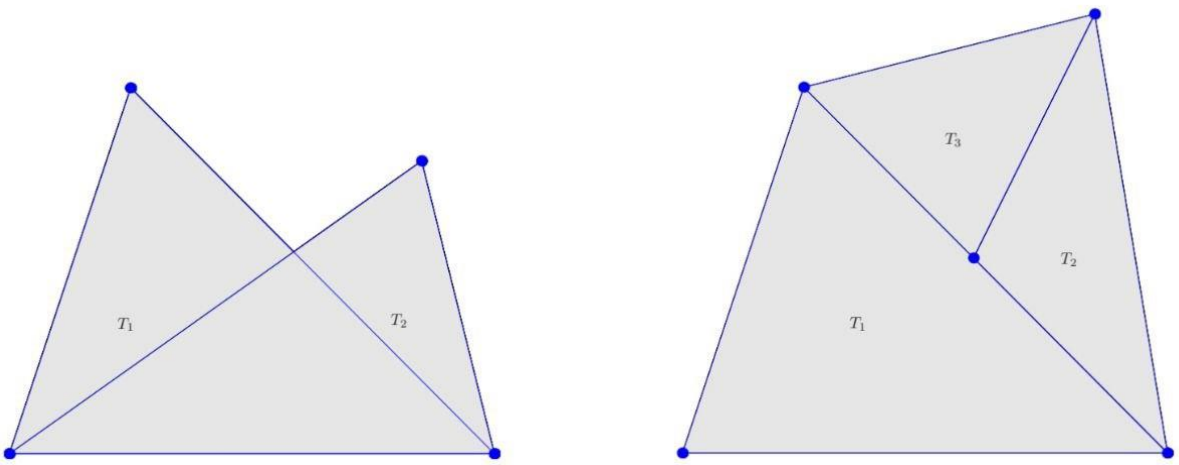


Figura 2 Triangolazioni proibite

Esempio di triangolazione ammissibile di $\bar{\Omega}$:

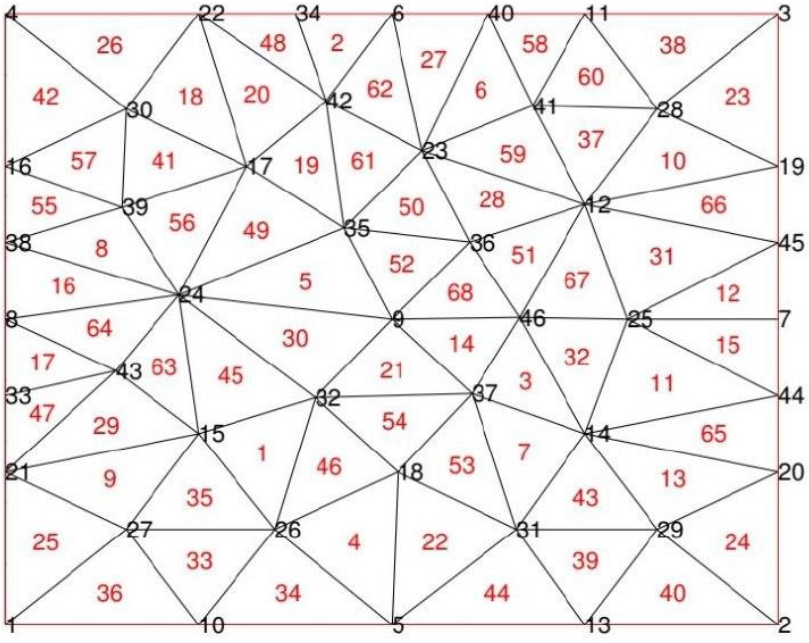


Figura 3 Triangolazione ammissibile

Sia τ la collezione dei triangoli in cui decomponiamo Ω; diremo che τ è una triangolazione di Ω. Dunque

$$\overline{\Omega} = \bigcup_{T \in \mathcal{T}} T \, .$$

Dato un triangolo $T \in \tau$, indichiamo con

$$h_T = \text{diam}(T)$$

il suo diametro, ossia la massima distanza tra i suoi punti (che coincide con la lunghezza del lato maggiore). Poniamo poi

$$h = \max_{T \in \mathcal{T}} h_T \, ;$$

tale parametro rappresenta una misura della finezza della triangolazione

I vertici dei triangoli di τ sono detti i nodi della triangolazione. Distinguiamo tra

- nodi interni, appartenenti a Ω, che supponiamo essere in numero di N_h^i,
- nodi di bordo, appartenenti a ∂Ω, che supponiamo essere in numero di N_h^b.

Il numero totale di nodi della triangolazione sarà dunque

$$\mathcal{N}_h = \mathcal{N}_h^i + \mathcal{N}_h^b \, .$$

Le equazioni che governano il comportamento fisico del sistema sono quindi formulate in termini di questi elementi e risolte simultaneamente per ottenere una soluzione approssimata dell'intero dominio. Nello specifico si riducono le equazioni differenziali alle derivate parziali che governano un dominio di interesse ad un sistema di equazioni algebriche.

Supponiamo che Γ_D sia unione di lati di triangoli. Supponiamo inoltre (per pura semplicità espositiva) che i nodi interni e di bordo su Γ_N siano numerati per primi rispetto a quelli su Γ_D . L'insieme degli spostamenti ammissibili discreti, o funzioni di forma discrete, viene definito come

$$V_h(g) = \{v_h \in \mathcal{V}_h \ : \ v_h(\boldsymbol{x}_j) = g(\boldsymbol{x}_j) \text{ per ogni } \boldsymbol{x}_j \in \Gamma_D\} \ ;$$

l'insieme delle variazioni ammissibili discrete, o funzioni test discrete, sarà dunque $\mathrm{V}_h(0)$.

Le funzioni di $\mathrm{V}_h(\mathrm{g})$ e di $\mathrm{V}_h(0)$ sono univocamente determinate dai loro valori nei nodi interni e appartenenti a Γ_N della triangolazione. Tali nodi sono dunque associati a dei gradi di libertà; indichiamo ancora con N il loro numero (con $\mathrm{N}_h^i \leq \mathrm{N} < \mathrm{N}_h$). Si avrà pertanto

$$V_h(0) = \text{vett}\,\{\varphi_j \ : \ 1 \leq j \leq N\} \ ,$$

mentre una funzione $u_h \in \mathrm{V}_h(\mathrm{g})$ sarà rappresentabile come

$$u_h(\boldsymbol{x}) = \sum_{k=1}^{N} u_k \varphi_k(\boldsymbol{x}) + \sum_{k=N+1}^{\mathcal{N}_h} g_k \varphi_k(\boldsymbol{x}) \ ,$$

avendo posto $g_k = g(\boldsymbol{x_k})$.

La formulazione variazionale discreta è allora la seguente:

$$\begin{cases} u_h \in V_h(g) \text{ e soddisfa} \\ \displaystyle\int_\Omega \mu \nabla u_h \cdot \nabla v_h \, d\boldsymbol{x} = \int_\Omega f v_h \, d\boldsymbol{x} + \int_{\Gamma_N} \psi \, v_h \, ds \qquad \text{per ogni } \ v_h \in V_h(0) \ . \end{cases}$$

Scegliendo di volta in volta $v_h = \varphi_j$, abbiamo immediatamente le N equazioni soddisfatte da u_h:

$$\int_\Omega \mu \nabla u_h \cdot \nabla \varphi_j \, d\boldsymbol{x} = \int_\Omega f \varphi_j \, d\boldsymbol{x} + \int_{\Gamma_N} \psi \, \varphi_j \, ds \, , \qquad 1 \leq j \leq N \, .$$

Sostituiamo l'espressione precedente per u_h a primo membro e portiamo a secondo membro quanto dipende dal dato g. Ponendo, come sempre,

$$a_{jk} = \int_\Omega \mu \nabla \varphi_k \cdot \nabla \varphi_j \, d\boldsymbol{x} \, ,$$

otteniamo il sistema algebrico

$$\sum_{k=1}^{N} a_{jk} u_k = \int_\Omega f \varphi_j \, d\boldsymbol{x} + \int_{\Gamma_N} \psi \, \varphi_j \, ds - \sum_{k=N+1}^{\mathcal{N}_h} a_{jk} g_k \, , \qquad 1 \leq j \leq N \, ,$$

che scriviamo ancora nella forma

$$Au = f$$

DERIVAZIONE DEL PROBLEMA RIDOTTO 2D-1D

Come già anticipato, l'obiettivo della trattazione è di analizzare un approccio numerico per la simulazione di equazioni ellittiche accoppiate bidimensionali e unidimensionali derivanti dalla riduzione della dimensione geometrica di problemi 2D-2D con inclusioni sottili, ovvero con una larghezza molto inferiore alla loro lunghezza.

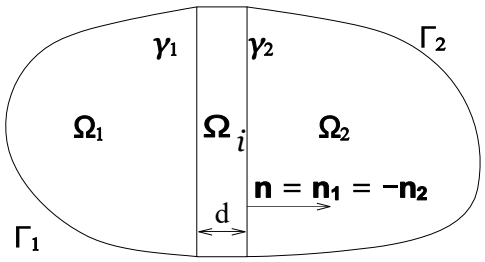
In questo paragrafo viene descritta la derivazione del problema 2D-1D a partire dal corrispondente problema equidimensionale 2D-2D.

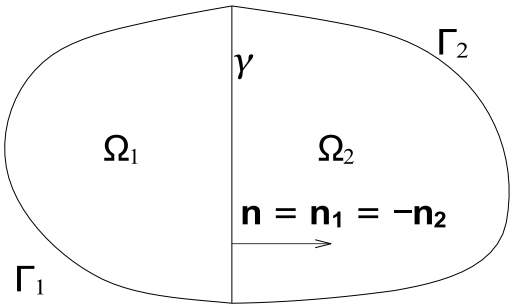
Supponiamo, dunque, che Ω sia un dominio convesso in R^n, $n = 2$, e denotiamo con $\Gamma = \partial\Omega$ il bordo di Ω. Supponiamo che in Ω sia posta un'equazione di deformazione elastica:

$$\begin{cases} -\nabla \cdot (\mu \nabla u) = f & \text{in } \Omega\,, \\ u = 0 & \text{su } \partial\Omega\,. \end{cases}$$

dove u è lo spostamento, μ il coefficiente di taglio, f una forzante, e u=0 è lo spostamento nullo sul bordo $\Gamma = \partial\Omega$. Supponiamo che μ sia strettamente positivo in Ω.

Supponiamo, come in figura, che l'inclusione Ω_i sia un sottodominio di Ω e che





ci sia un segmento γ e un vettore unitario n = n_1 = -n_2 normale a γ tale che

$$\Omega_i = \left\{x \in \Omega: x = s + rn \; per \; s \in \gamma \; e \; r \in \left(-\frac{d(s)}{2}, \frac{d(s)}{2}\right)\right\},$$

dove $d(s)$ denota lo spessore della inclusione in $s \in \gamma$.

Supponiamo anche che Ω_i separi Ω in due sottodomini connessi,

$$\Omega \backslash \overline{\Omega}_i = \Omega_1 \cup \Omega_2, \qquad \Omega_1 \cap \Omega_2 = 0.$$

Denotiamo con Γ_j la parte del bordo di Ω_j condivisa con il bordo di Ω, j = 1, 2, *i*,

$$\Gamma_j = \partial\Omega_j \cap \Gamma, \quad i = 1,2,i,$$

e denotiamo con γ_j la parte del bordo di Ω_j condivisa con il bordo della inclusione Ω_i, j = 1, 2,

$$\gamma_j = \partial\Omega_j \cap \partial\Omega_i \cap \Omega, \qquad i = 1,2,i,$$

Sia η il vettore normale unitario esterno su Γ.

Se denotiamo con u_j, μ_j e f_j le restrizioni di u, μ, e f, rispettivamente, a Ω_j, j = 1, 2, *i*, possiamo riscrivere il problema sopra come il seguente problema di trasmissione:

$$-\nabla(\mu\nabla u_j) = f_j \qquad in\ \Omega_j, \qquad j = 1,2,i$$

$$u_j = 0 \qquad in\ \Gamma_j \qquad j = 1,2,i$$

$$u_j = u_i \qquad in\ \gamma_j \qquad j = 1,2$$

$$\nabla u_j \cdot n_j = \nabla u_i \cdot n_j \qquad in\ \gamma_j \qquad j = 1,2$$

Il modello ridotto si ottiene mediando lungo i segmenti $[s - d(s)n, s + d(s)n], s \in \gamma$, normali a γ.

Possiamo dunque scrivere:

$$\nabla_t \tau_i + \nabla_n \tau_i = f_i \quad in\ \Omega_i$$

dove ∇_t è la divergenza nella direzione tangenziale e ∇_n la divergenza lungo la normale e $\tau_i = \mu\nabla u_i$.

Integrando lungo la direzione normale all'inclusione, otteniamo:

$$\tau_i \cdot n|_{\gamma 2} - \tau_i \cdot n|_{\gamma 1} + \nabla_t \tau_{1D} = f_i \quad in\ \gamma\ ,$$

Con $\tau_{1D}\ uguale\ a\ \int_{-d/2}^{d/2} \tau_{i,t} dn$,con τ_i soluzione lungo l'asse mediano dell'inclusione e $\tau_i \cdot n|_{\gamma 2} - \tau_i \cdot n|_{\gamma 1}$ che risulta essere un termine di sorgente addizionale.

Possiamo scrivere per il gradiente:

$$\tau_{i,t} = \mu \nabla_t u_i \; in \; \Omega_i$$

$$\tau_{i,n} = \mu \nabla_n u_i \; in \; \Omega_i$$

E integrando lungo la direzione normale all'inclusione abbiamo:

$$\tau_{1D} = \mu d \nabla_t u_{1D} \; in \; \gamma$$

Con $u_{1D} \; uguale \; a \; \frac{1}{d}\int_{-d/2}^{d/2} u_i dn$ e μ definita positiva nel medesimo intervallo.

Abbiamo così ricavato le equazioni in γ in un problema di deformazione elastica accoppiato.

Possiamo dunque scrivere le equazioni del modello ridotto in Ω e l'inclusione:

$$-\nabla(\mu \nabla u_{2D}) = f + \phi \delta_\gamma \; in \; \Omega$$

$$-\nabla(\mu \nabla u_{1D}) = f_i + \phi \;\; in \; \gamma$$

Dove ϕ rappresenta il salto della derivata co-normale:

$$\left(\nabla u \cdot n_\gamma\right)_2 - \left(\nabla u \cdot n_\gamma\right)_1$$

Scriviamo la formulazione variazionale nel dominio Ω. Si noti che non è possibile applicare il teorema della divergenza a tutto il dominio, ma bisogna dividerlo nelle due parti divise da γ.

$$\left(\int_\Omega \mu \nabla u_h \nabla v_h dx\right)_2 - \left(\int_{d\Omega} (\nabla u_h n) v_h dx\right)_2 + \left(\int_\Omega \mu \nabla u_h \nabla v_h dx\right)_1 - \left(\int_{d\Omega} (\nabla u_h n) v_h dx\right)_1$$
$$= \left(\int_\Omega f v_h dx\right)_2 + \left(\int_\Omega f v_h dx\right)_1 \; per \; ogni \; v_h \epsilon V_h$$

Considerando le ipotesi definite nel precedente paragrafo, ovvero la soluzione pari a 0 sul bordo del dominio e agli estremi dell'inclusione, e accorpando alcuni termini, l'equazione si può riscrivere:

$$\int_{\Omega} \mu \nabla u_h \nabla v_h dx \; - \left(\int_{\gamma} (\nabla u_h n_\gamma) v_h dx \; \right)_2 + \left(\int_{\gamma} (\nabla u_h n_\gamma) v_h dx \; \right)_1 = \int_{\Omega} f v_h dx \quad per\ ogni\ v_h \epsilon V_h$$

Poiché su γ, $u_1 = u_2\ per\ ipotesi\ e\ v_1 = v_2\ \ su\ \gamma$,possiamo scrivere:

$$\int_{\Omega} \mu \nabla u_h \nabla v_h dx \; - \int_{\gamma} \left[(\nabla u_h n_\gamma)_2 - (\nabla u_h n_\gamma)_1 \right] v_h dx \; = \int_{\Omega} f v_h dx \quad per\ ogni\ v_h \epsilon V_h$$

Il termine tra parentesi quadre risulta proprio essere il sopra citato salto della derivata co-normale. Per cui l'equazione si semplifica:

$$\int_{\Omega} \mu \nabla u_h \nabla v_h dx \; = \int_{\Omega} f v_h dx \; + \int_{\gamma} \phi \; v_h dx \quad per\ ogni\ v_h \epsilon V_h$$

È possibile riscrivere dunque tale equazione evidenziando la soluzione u_{2D} e scrivere la corrispondente equazione riferita all'inclusione.

$$\int_{\Omega} \mu_{2D} \nabla u_{2Dh} \nabla v_h dx \; = \int_{\Omega} f v_h dx \; + \int_{\gamma} \phi \; v_h dx \quad per\ ogni\ v_h \epsilon V_{2Dh}$$

$$\int_{\gamma} \mu_{1D} \nabla u_{1Dh} \nabla v_h dx \; = \int_{\gamma} f v_h dx \; + \int_{\gamma} \phi \; v_h dx \quad per\ ogni\ v_h \epsilon V_{1Dh}$$

Al fine di ottenere un metodo di decomposizione dei domini, introduciamo una variabile ausiliaria ψ che rappresenta la soluzione u_{2D} e u_{1D} su γ.

Questo vincolo viene tradotto nel funzionale quadratico J, definito come:

$$J = \frac{1}{2}(\left|\left|\psi - u_{2D}\right|\right|_{\gamma}^{2} + \left|\left| \psi - u_{1D}\right|\right|_{\gamma}^{2})$$

E che rappresenta l'errore nell'accoppiamento su γ con u_{2D} e u_{1D}.

Ciò vuol dire che se J=0 allora sia u_{2D} che u_{1D} sono pari a ψ su γ.

Perciò l'obiettivo è minimizzare J vincolato dal soddisfacimento delle equazioni alle derivate parziali, ovvero, trovare $u_{1D} \in V_{1D}(0)$ e $u_{2D} \in V_{2D}(0)$ che soddisfano:

$$\min J = \frac{1}{2}\left(\left|\left|\psi - u_{2D}\right|\right|_{\gamma}^{2} + \left|\psi - u_{1D}\right|_{\gamma}^{2}\right)$$

tale che

$$\int_{\Omega} \mu_{2D} \nabla u_{2Dh} \nabla v_h dx = \int_{\Omega} f v_h dx + \int_{\gamma} \phi v_h dx \quad per\ ogni\ v_h \epsilon V_{2Dh}$$

$$\int_{\gamma} \mu_{1D} \nabla u_{1Dh} \nabla v_h dx = \int_{\gamma} f v_h dx + \int_{\gamma} \phi v_h dx \quad per\ ogni\ v_h \epsilon V_{1Dh}$$

LA MATRICE DISCRETA

Il metodo esposto è vantaggioso perché si basa su una solida formulazione matematica e consente l'impiego di reti mesh che non devono necessariamente corrispondere alle interfacce tra i sottodomini. Infatti, l'uso del funzionale J per accoppiare le soluzioni permette di applicare mesh completamente distinte per ciascun dominio e per le variabili che agiscono come interfaccia, senza restrizioni teoriche sulla loro grandezza.

In questo paragrafo si descrive la versione discreta del problema matematico specificato nei paragrafi precedenti.

Per trattare il caso più generale possibile, fin dall'inizio si considera l'esistenza di molteplici segmenti che percorrono il dominio in esame. Si analizzano I segmenti di varia lunghezza e direzione, rappresentati come $\Lambda_i = \{\lambda_i(s), s \in (0, S_i)\}$, $i = 1,...,I$.

Si estende il dominio D a tutto lo spazio e si utilizza una mesh tetraedrica T per il dominio Ω, definendo su di essa le funzioni di base Lagrangiane degli elementi finiti $\{\varphi_k\}_{k=1}^{N}$, in modo che $U = \sum_{k=1}^{N} U_k \varphi_k$ rappresenti l'approssimazione discreta dello spostamento u.

Successivamente, si creano tre discretizzazioni indipendenti per ogni segmento di Λ_i, chiamate $\hat{T}_i$, τ_i^{ϕ} e τ_i^{ψ}, indipendenti sia tra loro sia dalla mesh T. Si definiscono le funzioni di base $\{\hat{\varphi}_{i,k}\}_{k=1}^{\hat{N}_i}$ su $\hat{T}_i$, $\{\theta_{i,k}\}_{k=1}^{N_i^{\phi}}$ su τ_i^{ϕ} e $\{\eta_{i,k}\}_{k=1}^{N_i^{\psi}}$ su τ_i^{ψ}, dove $\hat{N}_i$, N_i^{ϕ} e N_i^{ψ} indicano il numero di gradi di libertà (DOF) per le approssimazioni discrete delle variabili $\hat{U}_i$, $\overline{\phi}_i$, e $\hat{\Psi}_i$, rispettivamente:

$$\hat{U}_i = \sum_{k=1}^{\hat{N}_i} \hat{U}_{i,k} \hat{\varphi}_{i,k}, \quad \phi_i = \sum_{k=1}^{N_i^{\phi}} \phi_{i,k} \theta_{i,k}, \quad \psi_i = \sum_{k=1}^{N_i^{\psi}} \psi_{i,k} \eta_{i,k}$$

Definiamo quindi le seguenti matrici:

$$A \in R^{N\times N}\ s.t.\ (A)_{kl} = \int_{\Omega} \mu \nabla\varphi_k\, \varphi_l d\omega$$

$$\hat{A}_i \in R^{\hat{N}_i\times\hat{N}_i}\ s.t.\ (\hat{A}_i)_{kl} = \int_{\Lambda_i} \mu_i \mathrm{d} \frac{d\hat{\varphi}_{i,k}}{ds}\frac{d\hat{\varphi}_{i,l}}{ds} ds$$

$$B_i \in R^{N\times N_i^{\Phi}}\ s.t.\ (B_i)_{kl} = \int_{\Lambda_i} \varphi_{k|_{\Lambda_i}}\, \theta_{i,l} ds$$

$$\hat{B}_i \in R^{\hat{N}_i\times N_i^{\Phi}}\ s.t.\ (\hat{B}_i)_{kl} = \int_{\Lambda_i} \hat{\varphi}_{i,k}\, \theta_{i,l} ds$$

e i vettori:

$$f \in R^{N}\ s.t.\ f_k = \int_{\Omega} f\varphi_k\, d\omega$$

$$g_i \in R^{\hat{N}_i}\ s.t.\ (g_i)_k = \int_{\Lambda_i} dg_i \hat{\varphi}_{i,k}\, ds$$

Impostando $\hat{N} = \Sigma_{i=1}^{I}\hat{N}_i$, $N^{\psi} = \Sigma_{i=1}^{I}N^{\psi}{}_i$ e $N^{\phi} = \Sigma_{i=1}^{I}N^{\phi}{}_i$, possiamo raggruppare le matrici come segue per tutti i segmenti del dominio:

$$\boldsymbol{B} = [\boldsymbol{B_1}, \boldsymbol{B_2}, \ldots, \boldsymbol{B_I}] \in \mathbb{R}^{N\times N^{\phi}} \qquad \hat{\boldsymbol{B}} = \mathrm{diag}\left(\hat{\boldsymbol{B}}_{\boldsymbol{1}}, \ldots, \hat{\boldsymbol{B}}_{\boldsymbol{I}}\right) \in \mathbb{R}^{\hat{N}\times N^{\phi}}$$

e per segmenti non intersecanti abbiamo:

$$\hat{\boldsymbol{A}} = \mathrm{diag}\left(\hat{\boldsymbol{A}}_{\boldsymbol{1}}, \ldots, \hat{\boldsymbol{A}}_{\boldsymbol{I}}\right) \in \mathbb{R}^{\hat{N}\times\hat{N}},$$

Perciò possiamo scrivere:

$$\boldsymbol{A}U - \boldsymbol{B}\Phi = f$$
$$\hat{\boldsymbol{A}}\hat{U} + \hat{\boldsymbol{B}}\Phi = g$$

con

$$\hat{U} = \left[\hat{U}_1^T, \ldots, \hat{U}_{\mathcal{I}}^T\right]^T \in \mathbb{R}^{\hat{N}}; \quad g = [g_1^T, g_2^T, \ldots, g_{\mathcal{I}}^T]^T \in \mathbb{R}^{\hat{N}}$$
$$\Phi = \left[\Phi_1^T, \ldots, \Phi_{\mathcal{I}}^T\right]^T \in \mathbb{R}^{N^\phi}; \quad \Psi = \left[\Psi_1^T, \ldots, \Psi_{\mathcal{I}}^T\right]^T \in \mathbb{R}^{N^\psi}.$$

Al fine di avere una forma più compatta delle precedenti equazioni, imponiamo $W = (U, \hat{U})$ e

$$\mathcal{A} = \begin{bmatrix} \boldsymbol{A} & 0 \\ 0 & \hat{\boldsymbol{A}} \end{bmatrix}, \qquad \mathcal{B} = \begin{bmatrix} \boldsymbol{B} \\ -\hat{\boldsymbol{B}} \end{bmatrix}, \qquad \mathcal{F} = \begin{bmatrix} f \\ g \end{bmatrix},$$

Così l'equazione discreta diventa:

$$\mathcal{A}W - \mathcal{B}\Phi = \mathcal{F}$$

Per quanto riguarda il funzionale, definiamo le matrici:

$$\boldsymbol{G_i} \in \mathbb{R}^{N\times N} \text{ s.t. } (G_i)_{kl} = \int_{\Lambda_i} \varphi_{k|_{\Lambda_i}} \varphi_{l|_{\Lambda_i}} ds$$

$$\hat{\boldsymbol{G}}_{\boldsymbol{i}} \in \mathbb{R}^{\hat{N}_i\times\hat{N}_i} \text{ s.t. } (\hat{G}_i)_{kl} = \int_{\Lambda_i} \hat{\varphi}_{i,k}\, \hat{\varphi}_{i,l}\, ds$$

$$\boldsymbol{G}_{\boldsymbol{i}}^{\boldsymbol{\psi}} \in \mathbb{R}^{N_i^{\psi}\times N_i^{\psi}} \text{ s.t. } (G_i^{\psi})_{kl} = \int_{\Lambda_i} \eta_{i,k}\, \eta_{i,l}\, ds$$

$$\boldsymbol{C_i} \in \mathbb{R}^{N\times N_i^{\psi}} \text{ s.t. } (C_i)_{kl} = \int_{\Lambda_i} \varphi_{k|_{\Lambda_i}} \eta_{i,l}\, ds$$

$$\hat{\boldsymbol{C}}_{\boldsymbol{i}} \in \mathbb{R}^{\hat{N}_i\times N_i^{\psi}} \text{ s.t. } (\hat{C}_i)_{kl} = \int_{\Lambda_i} \hat{\varphi}_{i,k}\, \eta_{i,l}\, ds$$

E poi

$$\boldsymbol{G} = \sum_{i=1}^{\mathcal{I}} \boldsymbol{G}_i \in \mathbb{R}^{N\times N} \qquad \hat{\boldsymbol{G}} = \operatorname{diag}\left(\hat{\boldsymbol{G}}_{\boldsymbol{1}}^T, \ldots, \hat{\boldsymbol{G}}_{\boldsymbol{\mathcal{I}}}^T\right) \in \mathbb{R}^{\hat{N}\times\hat{N}} \qquad \boldsymbol{\mathcal{G}} = \begin{bmatrix} \boldsymbol{G} & 0 \\ 0 & \hat{\boldsymbol{G}} \end{bmatrix}$$

$$\boldsymbol{G}^{\boldsymbol{\psi}} = \operatorname{diag}\left(\boldsymbol{G}_{\boldsymbol{1}}^{\boldsymbol{\psi}}, \ldots, \boldsymbol{G}_{\boldsymbol{\mathcal{I}}}^{\boldsymbol{\psi}}\right) \in \mathbb{R}^{N^{\psi}\times N^{\psi}}$$

$$\boldsymbol{C} = [\boldsymbol{C_1}, \boldsymbol{C_2}, \ldots, \boldsymbol{C_{\mathcal{I}}}] \in \mathbb{R}^{N\times N^{\psi}} \qquad \hat{\boldsymbol{C}} = \operatorname{diag}\left(\hat{\boldsymbol{C}}_{\boldsymbol{1}}, \ldots, \hat{\boldsymbol{C}}_{\boldsymbol{\mathcal{I}}}\right) \in \mathbb{R}^{\hat{N}\times N^{\psi}} \qquad \boldsymbol{\mathcal{C}} = \begin{bmatrix} \boldsymbol{C} \\ \hat{\boldsymbol{C}} \end{bmatrix}.$$

Il funzionale discreto così si legge:

$$\tilde{J} = \frac{1}{2}\left(U^T \boldsymbol{G} U - U^T \boldsymbol{C}\Psi - \Psi^T \boldsymbol{C}^T U + \hat{U}^T \hat{\boldsymbol{G}} \hat{U} - \hat{U}^T \hat{\boldsymbol{C}}\Psi - \Psi^T \hat{\boldsymbol{C}}^T \hat{U} + 2\Psi^T \boldsymbol{G}^{\boldsymbol{\psi}}\Psi\right) =$$
$$= \frac{1}{2}\left(W^T \boldsymbol{\mathcal{G}} W - W^T \boldsymbol{\mathcal{C}}\Psi - \Psi^T \boldsymbol{\mathcal{C}}^T W + 2\Psi^T \boldsymbol{G}^{\boldsymbol{\psi}}\Psi\right).$$

Per cui la formulazione della matrice discreta del problema 2D-1D prende la forma:

$$\min_{(\Phi,\Psi)} \tilde{J}(\Phi, \Psi)$$

Le condizioni ottimali del primo ordine per il problema sopra corrispondono al sistema di punto di sella:

$$\mathcal{S} = \begin{bmatrix} \mathcal{G} & \mathbf{0} & -\mathcal{C} & \mathcal{A}^T \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathcal{B}^T \\ -\mathcal{C}^T & \mathbf{0} & 2G^{\Psi} & \mathbf{0} \\ \mathcal{A} & \mathcal{B} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$\mathcal{S} \begin{bmatrix} W \\ \Phi \\ \Psi \\ -P \end{bmatrix} = \begin{bmatrix} \mathcal{F} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

LA PROGRAMMAZIONE ORIENTATA AGLI OGGETTI

Nello studio in questione, per una programmazione innovativa e versatile, è stata utilizzata e approfondita la programmazione orientata agli oggetti (OOP). Quest'ultima rappresenta un paradigma di programmazione che consente agli sviluppatori di strutturare i loro software come collezioni di oggetti che interagiscono tra loro. Questo paradigma si basa su alcuni principi fondamentali, tra i quali ci sono incapsulamento ed ereditarietà, particolarmente utilizzati in tale lavoro. MATLAB, un ambiente di programmazione e linguaggio di scripting ampiamente utilizzato per il calcolo numerico, l'analisi dei dati, e la visualizzazione, supporta pienamente l'OOP, offrendo agli utenti un potente strumento per creare applicazioni complesse e modulari.

Fondamenti dell'OOP in MATLAB

Classe

La classe è il concetto centrale dell'OOP in MATLAB. Una classe definisce una nuova tipologia di dato che incapsula dati e funzioni specifiche (metodi) relative a quel tipo di dato. Le classi sono definite in MATLAB tramite file di classe, con la parola chiave classdef. Ogni classe può avere proprietà (variabili specifiche della classe) e metodi (funzioni specifiche della classe).

Esempio applicativo presente nel codice affrontato:

```
classdef DomainPoint < matlab.mixin.Copyable
   properties
      Id
      Coords
      marker
   end
   properties (Access=private)
      markerD=0;
      markerN=0;
      markerR=0;
   end
```

```
    methods
        function obj=DomainPoint(id,pt)
            if nargin>0
                obj.Id=id;
                obj.Coords=pt;
            end
            obj.marker=0;
        end
    end
end
```

Oggetto

Un oggetto è un'istanza di una classe. Dopo aver definito una classe, è possibile creare oggetti (istanze) di quella classe. Questo si fa chiamando il costruttore della classe, che è un metodo con lo stesso nome della classe.

Inizializzazione del Dominio R con la classe Domain2D:

```
R=Domain2D([0 -1;1 -1; 1 1;0 1]);
R.Initialize;
```

Ereditarietà

L'ereditarietà è un meccanismo dell'OOP che permette di definire una nuova classe basata su una classe esistente. La nuova classe, chiamata sottoclasse, eredita i metodi e le proprietà della classe esistente, chiamata superclasse, consentendo di riutilizzare, estendere o modificare il comportamento della superclasse o classe madre.

La classe Domain2d è figlia della classe Domain

```
classdef Domain2D < Domain
    properties
        inclusion=LineSeg;
        nI=0;
        BC
        Data
        sol
```

```
        opt
        Npsi
        Nlambda
        Nu
    end
    methods
        function obj=addInclusion(obj,L)
            obj.nI=obj.nI+1;
            obj.inclusion(obj.nI)=L;
        end
        …
    end
end
```

Nel lavoro presentato, è di rilevante importanza sottolineare che, tutte le classi sono sottoclassi della classe madre ‘matlab.mixin.Copyable’, la quale è una superclasse handle astratta che permette di eseguire copie profonde. Ciò vuol dire che permette di creare nuovi oggetti e simultaneamente creare copie di tutti gli oggetti a cui si riferiscono le proprietà dell’oggetto originale. In tal modo, ogni copia creata non condividerà alcun riferimento agli oggetti con l’originale.

Incapsulamento

L'incapsulamento è il principio di nascondere i dettagli interni di implementazione di una classe e di esporre solo le funzionalità che sono sicure e necessarie per l'uso esterno. In MATLAB, è possibile definire le proprietà e i metodi di una classe come public, protected, o private, a seconda del livello di accesso desiderato.

TEST NUMERICI

Definito, nei precedenti paragrafi, il modello matematico di studio, si è proceduto all'implementazione su Matlab di un codice che applicasse tale metodo. Nello specifico, il primo passo è stato quello di testare tale codice ad un caso di un dominio rettangolare con una sola inclusione con un problema di cui si conosce la soluzione esatta (vedi cap. Test 1 inclusione). Successivamente, è stato affrontato un problema che considerasse la presenza di più inclusioni parallele tra loro, nello specifico 4 (vedi cap. Test 4 inclusioni), prendendo un caso di applicazione realistica quale il vetro resina composto da una matrice di poliestere e delle fibre di vetro. In questo caso, il metodo è stato validato confrontando i risultati ottenuti rispetto alla soluzione del problema equidimensionale, ovvero il caso in cui sia le fibre che la matrice in cui esse sono immerse siano rappresentate da domini 2D, risolto con elementi finiti standard e per cui è stato sviluppato un codice dedicato, sempre in ambiente Matlab. Nei paragrafi a seguire si mostrano, dunque, i risultati raggiunti.

Test 1 inclusione (caso bi-dimensionale 2D-1D)

Il primo test è stato basato su un dominio 2D dato da un rettangolo di dimensioni $L_x = 1\ e\ L_y = 2$ con un vertice in (0; -1) e una inclusione con origine in (0;0), perpendicolare all'asse y e lunga $L_x = 1$.

Tale test è stato utile ai fini della verifica del corretto funzionamento del codice Matlab poiché del problema in questione si conoscevano le soluzioni esatte 2D e 1D.

La spiegazione delle varie sezioni del codice è rimandata al paragrafo 'Test 4 inclusioni (caso bi-dimensionale 2D-1D)'.

La diffusività sia della matrice che dell'inclusione è pari a 1, la forzante costante pari a 2, due lati opposti con condizioni di Neumann non omogenea pari a 1 e due lati con condizione di Dirichlet non omogenea data dalla funzione |y|+(x(1-x)).

Nel seguente grafico sono state dunque rappresentate sia la soluzione 2D del problema bidimensionale che la soluzione esatta data dalla funzione |y|+(x(1-x)).

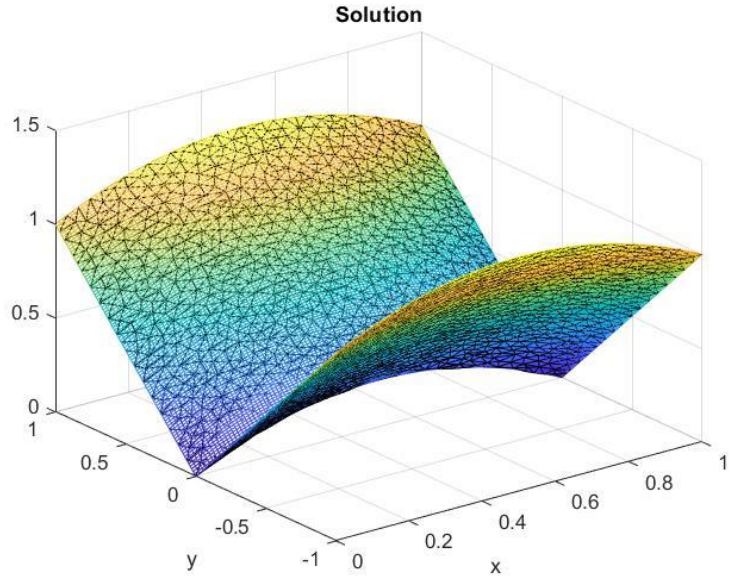


Figura 4 Soluzione problema accoppiato di cui si conosce soluzione esatta

Al fine di una corretta valutazione dei risultati ottenuti si è deciso di calcolare l'errore massimo tra la soluzione U1D e la soluzione esatta y=x(1-x) per cinque mesh differenti con gradi di libertà differenti (vedi tab.1)

$$\|E\|_\infty = \max_{x \in I} |E(x)|$$

$$E(x) = U_{1D} - U_{ex} \quad con\ U_{ex}\ soluzione\ esatta.$$

Nel grafico e nella tabella seguenti sono mostrati i risultati sottolineando i gradi di libertà di ogni mesh presa in considerazione.

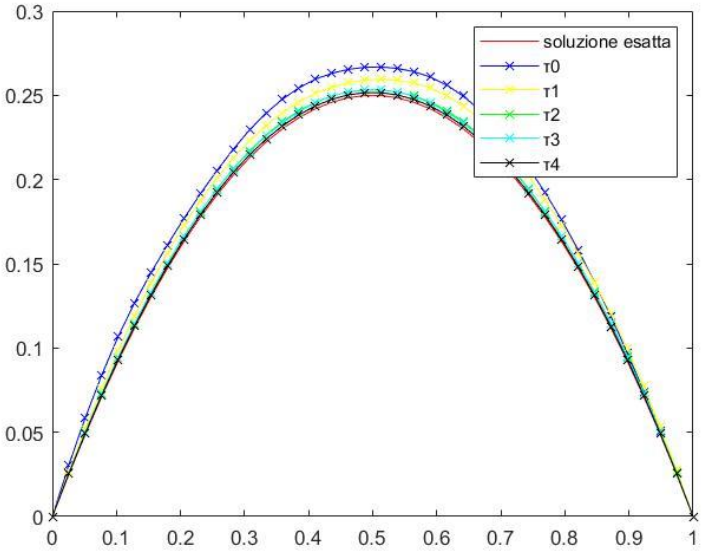


Figura 5 Confronto soluzione esatta e soluzion1 U1D per le 5 mesh

Triangolazione	Ndof	Errore
τ0	143	0.0194
τ1	305	0.0135
τ2	1553	0.0043
τ3	3107	0.0036
τ4	15.500	0.0017

Tabella 1 Errore in norma infinito per le 5 differenti mesh

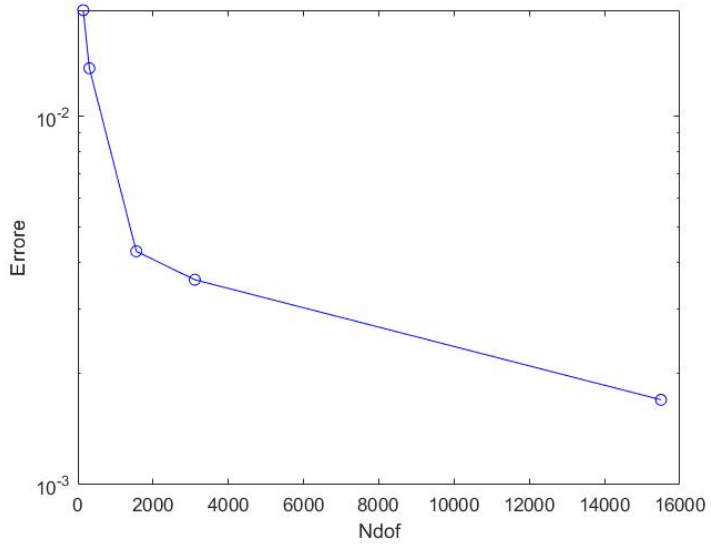


Figura 6 Andamento dell'errore al variare dei gradi di libertà (scala semi-logaritmica)

Come ci si aspettava, l'errore decresce all'aumentare del numero di gradi di libertà, ovvero al raffinamento della mesh, confermando la validità del metodo proposto.

Test 4 inclusioni (caso equidimensionale 2D-2D)

Come anticipato nell'introduzione ai test numerici, per il problema che prevede 4 inclusioni è stato trattato anche il modello equidimensionale (2D-2D) al fine di validare il metodo ridotto (2D-1D). In questa parte si affronta il modello equidimensionale. Esso utilizza una libreria ad oggetti Matlab (Bardella, 2021), la quale permette in maniera semplificata di creare domini con differenti proprietà e mesh. Come anticipato l'esempio prevede:

- un Dominio 2D che rappresenta nell'esempio specifico il poliestere. Esso è un rettangolo di dimensioni $L_x = 1\ e\ L_y = 2$ con un vertice in (0; -1).
- 4 Domini 2D che rappresentano le 4 fibre di vetro, le quali hanno $L_x = 1\ e\ L_y = 0.01$, tutte con linee di mezzeria parallele tra loro, con origine rispettivamente in (0; -7), (0; -1), (0; 3), (0; 8).

Il codice si può dividere, per una più semplice spiegazione, nelle seguenti 6 parti:

1. **Inizializzazione**:

   Definisce i coefficienti per i due materiali distinti, che rappresentano la matrice e la fibra nel composito. Mu_matrice e mu_fibra rappresentano i coefficienti di taglio del poliestere ($\mu = 1428\,, E = 4000\ Mpa, \nu = 0.4$) e della fibra di vetro ($\mu = 29600\,, E = 74000\ Mpa, \nu = 0.25$). Si noti che per la teoria dei coefficienti effettivi e reali, se consideriamo uno spessore di larghezza 0.01, come già anticipato, il mu _fibra nel caso del problema ridotto deve essere riscalato con lo spessore dell'inclusione. Ciò accade perché nel passaggio al caso in cui la fibra sia monodimensionale, ovvero rappresentata da un dominio 1D (e che abbia i coefficienti sopra introdotti), lo spessore della fibra compare a seguito delle operazioni di media lungo la direzione normale.

2. **Geometria**:

   In questa parte si definisce la geometria del sistema. Si utilizza la funzione regions.rectN per definire rettangoli che rappresentano la geometria del dominio del problema. Il dominio è suddiviso in cinque regioni color blu per la matrice (R1, R2, R3, R4, R5) e quattro strisce color rosso che rappresentano le fibre (FIBRA1, FIBRA2, FIBRA3, FIBRA4). Queste regioni sono poi aggregate per formare il dominio totale, diviso in MATRICE e FIBRA.

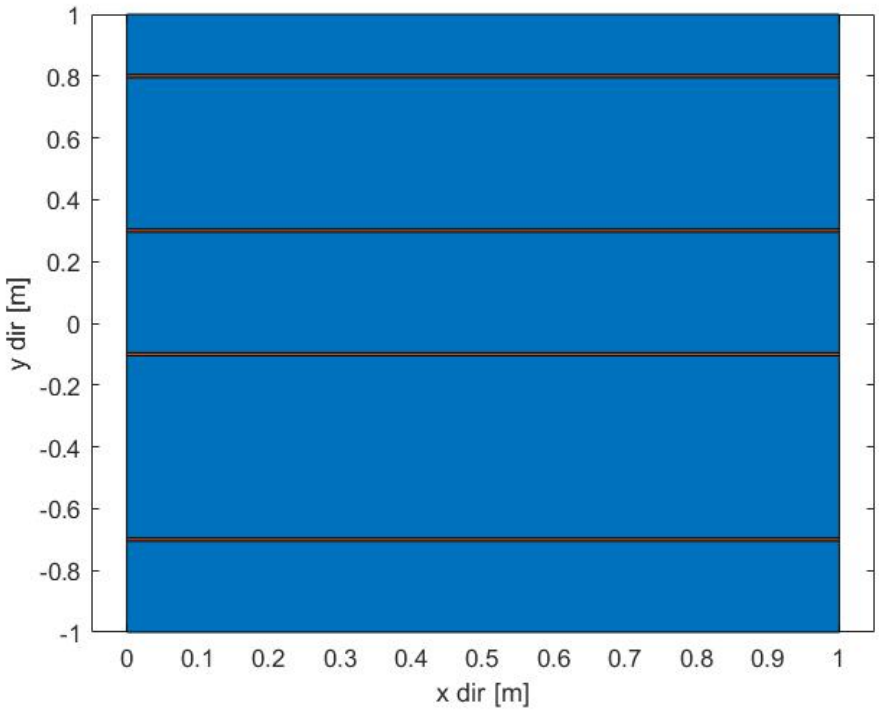


Figura 7 Dominio 2D. In blu la matrice di poliestere, in rosso le fibre di vetro.

3. **Assegnazione dei materiali**:

   Vengono, così, assegnati i coefficienti di taglio (mu) ai rispettivi domini (MATRICE e FIBRA).

4. **Definizione della forzante**:

   Definisce una funzione *f(x,y)* che indica la forzante nel problema. Nel caso specifico è *f*=2. Questa funzione, in un caso più generale, restituisce valori basati sulla posizione (x,y). In questo specifico, invece, non risulta dipendente dalla posizione.

5. **Condizioni al bordo**:

Specifica le condizioni al contorno per i bordi dei domini. Viene utilizzata la funzione boundaries per definire vari tipi di condizioni al contorno. In questo caso le condizioni sono di Dirichlet omogeneo in x=0 e x=1, Neumann omogeneo in y=-1 e y=1. Nell'immagine, le condizioni di Neumann son in blu tratteggiato e quelle di Dirichlet sono in giallo.

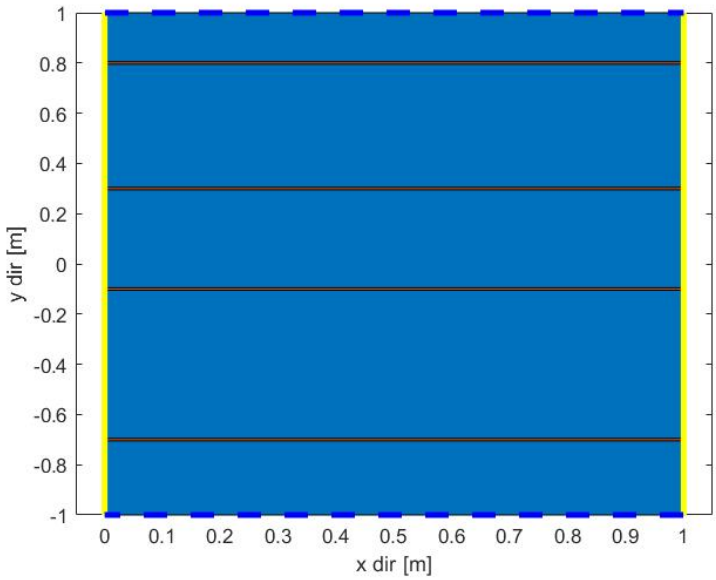


Figura 8 Condizioni al contorno. Condizioni di Dirichlet in giallo, condizioni di neumann In blu tratteggiato.

6. **Mesh**:

Dunque, si passa alla generazione di una mesh 2D per il dominio definito, con il metodo mesh2D(dominio, *delta*), dove *delta* indica la dimensione massima dell'elemento della mesh. *Delta* può avere valori diversi per la matrice e per le fibre. In questo caso si è posto $delta\_{fibre} = 1e-5$ e $delta\_{matrice} = 1e-3$.

Le immagini che seguono mostrano la mesh così ottenuta.

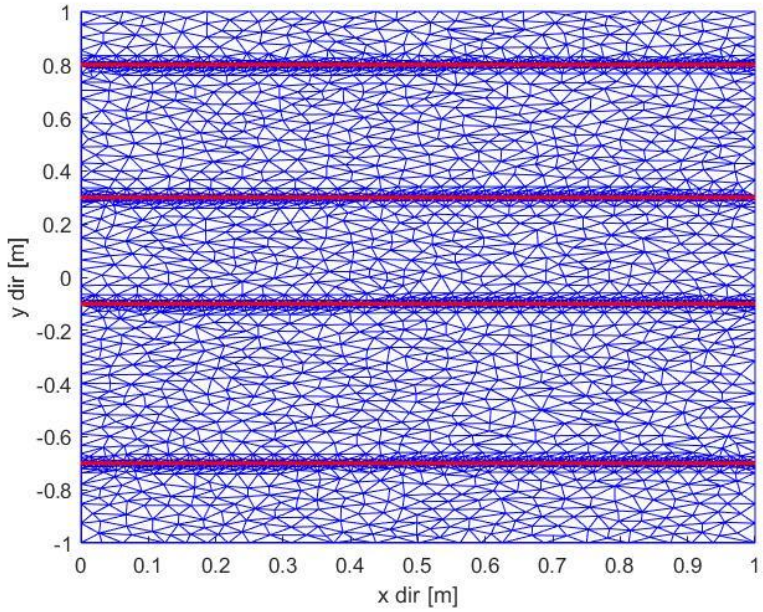


Figura 9 Mesh 2D per il problema equidimensionale nel caso del vetro resina in presenza di 4 fibre di vetro parallele.

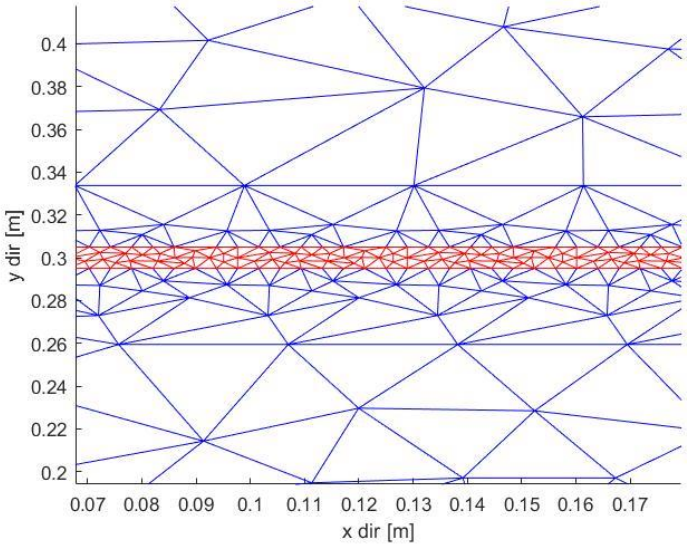


Figura 10 Zoom su Mesh 2D della regione FIBRA e della MATRICE

7. **Soluzione del problema**:

Si costruisce la matrice del sistema e il vettore termine noto (A, b) basandosi sulla mesh e sulla forzante *f*, e poi risolve il sistema lineare per ottenere i valori

dei gradi di libertà (U_dof). Successivamente, i risultati vengono visualizzati in tutti i nodi della mesh.

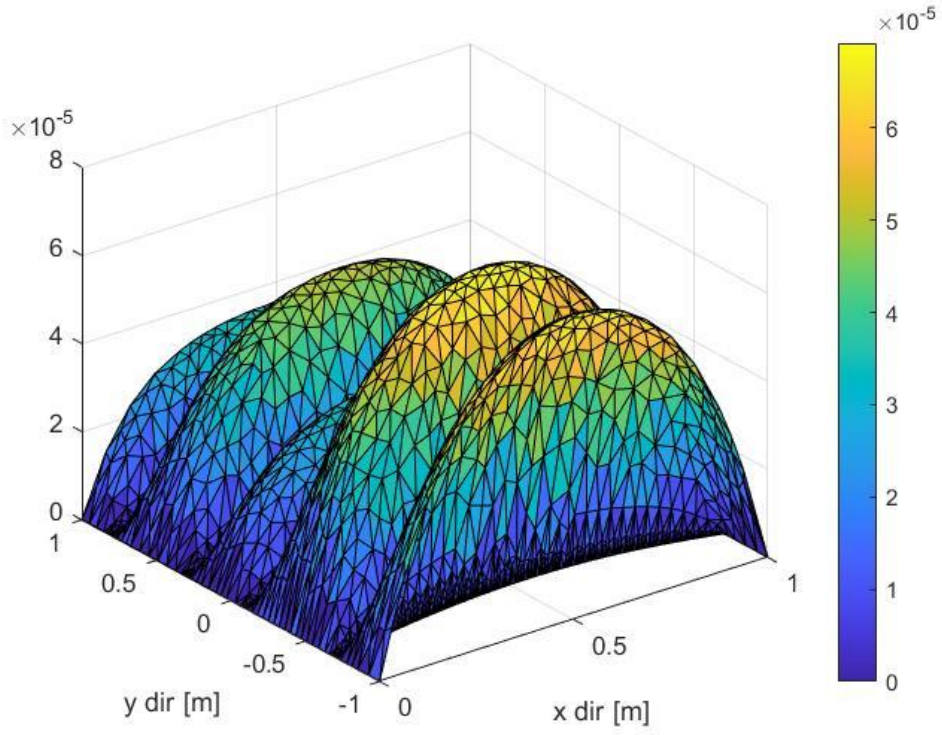


Figura 11 Soluzione del modello equidimensionale per il caso del vetroresina .Matrice di poliestere e 4 fibre di vetro.

Questo script richiede, come già in precedenza sottolineato, accesso a una libreria MATLAB che implementa le funzioni utilizzate (come regions.rectN, boundaries, mesh2D, buildMatriceETermineNoto, ecc.), le quali sono specifiche di tale toolbox personalizzato per la modellazione agli elementi finiti (Bardella, 2021).

A seguire viene illustrato il codice di interesse:

```
%%GENERAL_EQUIDIMENSIONALPROBLEM

mu_matrice=1428;
mu_fibra=296000; %coeff effetivi e coef reali ,mu_reale=mu_effettiva/spessore
```

```
%GEOMETRIA
yy=0.005; %semi-larghezza fibra
R1=regions.rectN([0,-1],[1,-0.7-yy],'MeshMaxArea',1e-3);
R2=regions.rectN([0,-0.7+yy],[1,-0.1-yy]);
R3=regions.rectN([0,-0.1+yy],[1,0.3-yy]);
R4=regions.rectN([0,0.3+yy],[1,0.8-yy]);
R5=regions.rectN([0,0.8+yy],[1,1]);
MATRICE=R1+R2+R3+R4+R5;
FIBRA1=regions.rectN([0,-0.7-yy],[1,-0.7+yy]);
FIBRA2=regions.rectN([0,-0.1-yy],[1,-0.1+yy]);
FIBRA3=regions.rectN([0,0.3-yy],[1,0.3+yy]);
FIBRA4=regions.rectN([0,0.8-yy],[1,0.8+yy]);
FIBRA=FIBRA1+FIBRA2+FIBRA3+FIBRA4;
dominio(1)=MATRICE;
dominio(2)=FIBRA;
dominio(1).mu=mu_matrice;
dominio(2).mu=mu_fibra;
f=@(x,y) 2*(y<-0.7-yy | (y>-0.7+yy & y<-0.1-yy) |(y>-0.1+yy & y<0.3-yy)
|(y>0.3+yy & y<0.8-yy) | y>0.8+yy )+0*x+0*y;
% dominio.draw


%CONDIZIONI AL BORDO
dominio(1).Borders(1).Bc(1)= boundaries.none;
dominio(1).Borders(1).Bc(2)= boundaries.dirichlet(0);
dominio(1).Borders(1).Bc(3)= boundaries.neumann(0);
dominio(1).Borders(1).Bc(4)= boundaries.dirichlet(0);
```

```
dominio(1).Borders(2).Bc(1)= boundaries.none;
dominio(1).Borders(2).Bc(2)= boundaries.dirichlet(0);
dominio(1).Borders(2).Bc(3)= boundaries.none;
dominio(1).Borders(2).Bc(4)= boundaries.dirichlet(0);
dominio(1).Borders(3).Bc(1)= boundaries.none;
dominio(1).Borders(3).Bc(2)= boundaries.dirichlet(0);
dominio(1).Borders(3).Bc(3)= boundaries.none;
dominio(1).Borders(3).Bc(4)= boundaries.dirichlet(0);
dominio(1).Borders(4).Bc(1)= boundaries.none;
dominio(1).Borders(4).Bc(2)= boundaries.dirichlet(0);
dominio(1).Borders(4).Bc(3)= boundaries.none;
dominio(1).Borders(4).Bc(4)= boundaries.dirichlet(0);
dominio(1).Borders(5).Bc(1)= boundaries.neumann(0);
dominio(1).Borders(5).Bc(2)= boundaries.dirichlet(0);
dominio(1).Borders(5).Bc(3)= boundaries.none;
dominio(1).Borders(5).Bc(4)= boundaries.dirichlet(0);
dominio(2).Borders(1).Bc(3)= boundaries.none;
dominio(2).Borders(1).Bc(2)= boundaries.dirichlet(0);
dominio(2).Borders(1).Bc(1)= boundaries.none;
dominio(2).Borders(1).Bc(4)= boundaries.dirichlet(0);
dominio(2).Borders(2).Bc(3)= boundaries.none;
dominio(2).Borders(2).Bc(2)= boundaries.dirichlet(0);
dominio(2).Borders(2).Bc(1)= boundaries.none;
dominio(2).Borders(2).Bc(4)= boundaries.dirichlet(0);
dominio(2).Borders(3).Bc(3)= boundaries.none;
dominio(2).Borders(3).Bc(2)= boundaries.dirichlet(0);
dominio(2).Borders(3).Bc(1)= boundaries.none;
```

```
dominio(2).Borders(3).Bc(4)= boundaries.dirichlet(0);
dominio(2).Borders(4).Bc(3)= boundaries.none;
dominio(2).Borders(4).Bc(2)= boundaries.dirichlet(0);
dominio(2).Borders(4).Bc(1)= boundaries.none;
dominio(2).Borders(4).Bc(4)= boundaries.dirichlet(0);
% dominio.draw('edge')
% dominio.draw('bc')


%MESH

Me=mesh2D(dominio,1e-5);


%soluzione
[A,b] = buildMatriceETermineNoto(Me, f);
U_dof = A\b;
U = Me.copyToAllNodes(U_dof);
Me.draw(U)
```

Test 4 inclusioni (caso ridotto 2D-1D)

In questo capitolo è affrontato il medesimo caso considerato nel paragrafo precedente, ovvero lo studio di 4 inclusioni (questa volta 1D) parallele tra loro, non equidistanti, in un dominio 2D rettangolare, coincidente con quello del caso equidimensionale.

L'esempio trattato è riconducibile ad un materiale composito, come ad esempio il vetro resina, il quale presenta delle fibre di vetro in una matrice di poliestere. Le sue proprietà al pari di quanto fatto per il 'Test 4 inclusioni (caso equidimensionale 2D-2D)', sono utilizzate ai fini del calcolo di una soluzione realistica.

A seguire, una spiegazione dettagliata di ciascuna parte del codice, che è stato diviso in sezioni per una più semplice interpretazione.

Per ulteriori approfondimenti riguardo le classi e il restante codice implementato su Matlab si può fare riferimento all'Appendice A.

Definizione del dominio:

Viene innanzitutto creato un dominio bidimensionale R definendo i punti che formano un rettangolo. Questo è un oggetto della classe Domain2D (vedi Appendice A) e i punti passati come argomento rappresentano i vertici del dominio.

Inizializzazione delle opzioni di mesh:

Viene definita la densità e il tipo di mesh per diverse variabili (psi, lambda, u) che rappresentano rispettivamente la soluzione dell'interfaccia, il valore del flusso sull'interfaccia e la soluzione del problema 1D. Le opzioni includono il numero di punti nella mesh (MeshN) e l'ordine degli elementi della mesh.

Creazione delle inclusioni:

Vengono create diverse linee (LineSeg) che rappresentano le inclusioni all'interno del dominio, con le loro posizioni, direzioni e le opzioni di mesh specificate in precedenza.

Queste linee sono aggiunte al dominio R tramite il metodo addInclusion della classe Domain2D.

Proprietà dei materiali:
Viene definito un vettore mu che contiene i valori dei coefficienti di taglio per ogni inclusione, specificamente per la fibra di vetro. Nello specifico con un ciclo for è stato attribuito un coefficiente di taglio a ciascuna inclusione.

Definizione del problema e delle condizioni al contorno:
Viene istanziato un oggetto ProblemData per definire la funzione forzante, la diffusività (legata al materiale del dominio, qui il poliestere).

Le condizioni al contorno vengono impostate usando la classe BoundaryConditions , che permette di specificare condizioni di Dirichlet e Neumann su vari bordi del dominio e delle inclusioni.

Mesh del dominio e delle inclusioni:
Viene calcolata la mesh del dominio e delle inclusioni con una dimensione specificata per gli elementi.

Costruzione della matrice di rigidezza e del vettore dei termini noti:
Viene calcolata la matrice di rigidezza A2D e il vettore b2D per il problema nel dominio 2D.

Vengono calcolate tutte le matrici e vettori aggiuntivi per gestire le interazioni tra il problema 2D e le inclusioni 1D, tra cui le matrici B2D, G2D, G1D, ecc., che rappresentano le varie matrici della formulazione matematica sopra sviluppata.

Assemblaggio del sistema globale e risoluzione:
Il sistema di equazioni lineari complessivo viene assemblato in una matrice M e risolto rispetto a bM per ottenere la soluzione sol. La soluzione per il problema 2D, u2D viene estratta da sol e mostrata nella figura che segue:

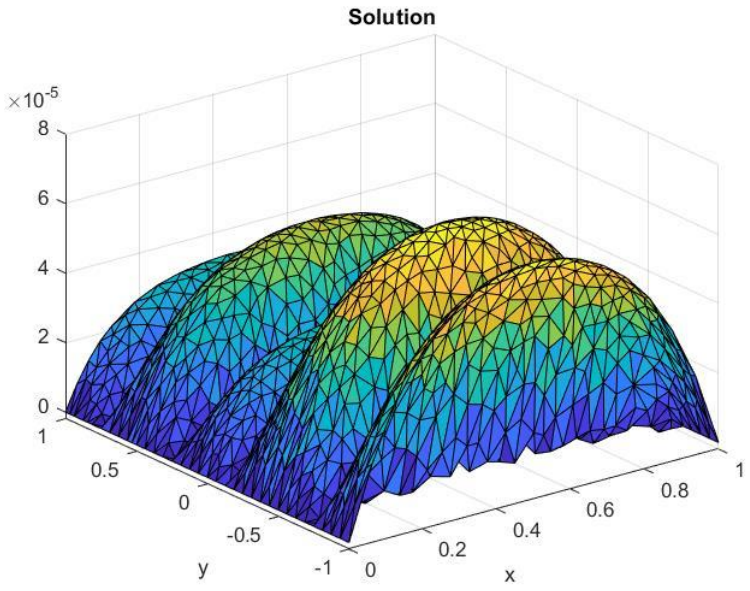


Figura 12 Soluzione per il problema accoppiato 1D-2D nel caso di vetroresina.

Inoltre, sono stati stampati anche i plot delle soluzioni relative alle 4 inclusioni presenti nel problema di riferimento.

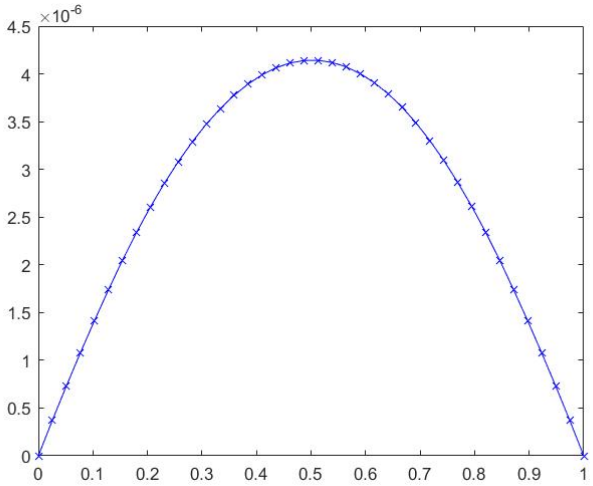


Figura 13 Soluzione per la fibra con vertici (-0,7;0) (-0,7;1)

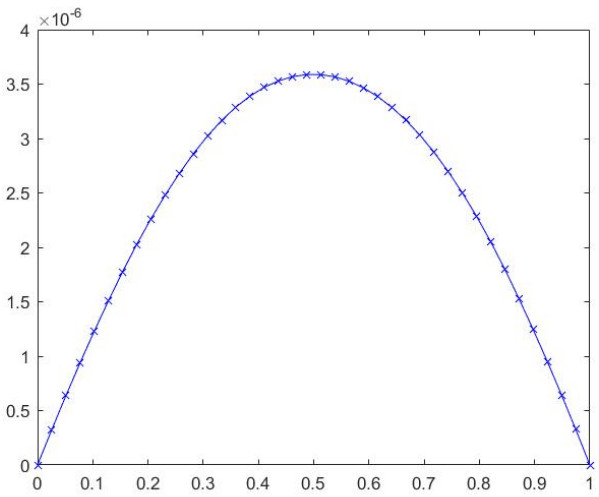


Figura 14 Soluzione per la fibra con vertici (-0,1;0) (-0,1;1)

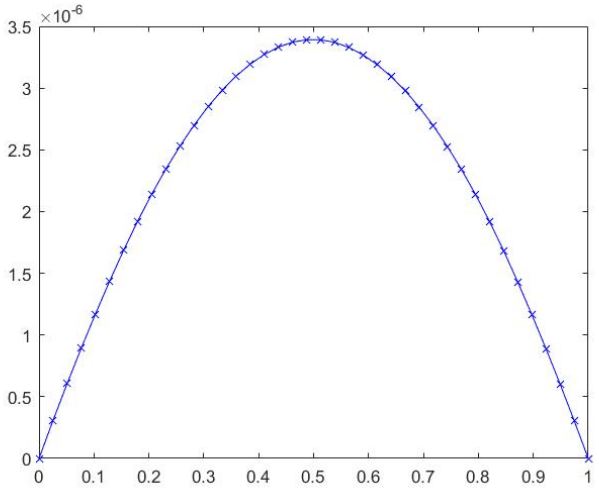


Figura 15 Soluzione per la fibra con vertici (0,3;0) (0,3;1)

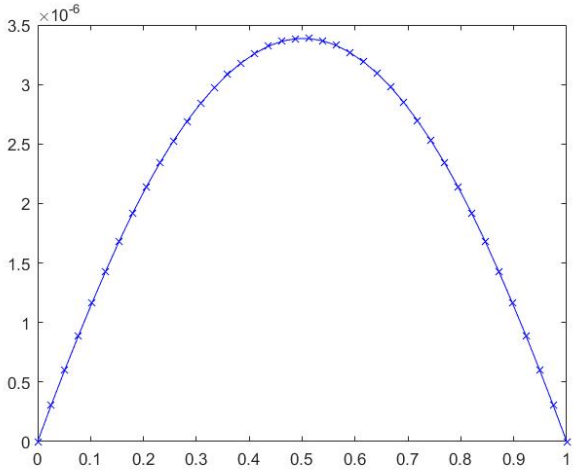


Figura 16 Soluzione per la fibra con vertici (0,8;0) (0,8;1)

A seguire viene illustrato il codice di riferimento:

```
%% GENERAL TEST

R=Domain2D([0 -1;1 -1; 1 1;0 1]);
R.Initialize;


nn=4;
opt.psiMeshType='EquallySpaced'; % psi= valore soluzione interfaccia - VARIABILE
PSI
opt.psiMeshN=10*nn/2;
opt.psiOrder=1;
opt.lambdaMeshType='EquallySpaced'; % lambda= valore flusso interfaccia - variable
PHI
opt.lambdaMeshN=8*nn/4;
opt.lambdaOrder=0;
opt.uMeshType='EquallySpaced'; % soluzione problema 1D. - VARIABILE uF
```

```
opt.uMeshN=10*nn;
opt.uOrder=1;
opt.uMeshBC=[1 1 0;opt.uMeshN 1 0];



L(1)=LineSeg([0; -0.7],[1; 0],1,opt);
L(2)=LineSeg([0; -0.1],[1; 0],1,opt);
L(3)=LineSeg([0; 0.3],[1; 0],1,opt);
L(4)=LineSeg([0; 0.8],[1; 0],1,opt);
mu=[29600; 29600; 29600; 29600]; %E/(2*(1+v))

for i=1:length(L)
   R.addInclusion(L(i));
end

n=R.nI; %n inclusion

Data=ProblemData;
Data.setForcing(@(x,y) 2+3*x+4+y);
Data.setDiffusivity(1428);  %  E=4000  v=0.4  POLIESTERE  (RESINA
TERMOIDURENTE)
Data.setGamma(0);

BC=BoundaryConditions;
BC.setBoundaryConditions(R.Points(1:4),'Dirichlet', 0);
BC.setBoundaryConditions(R.Edges([2 4]),'Dirichlet',0);
BC.setBoundaryConditions(R.Edges(1),'Neumann',0);
```

```
BC.setBoundaryConditions(R.Edges(3),'Neumann',0);

R.setBC(BC);
R.setData(Data);

R.computeMesh(0.001);
% R.mesh.draw_matlab;
% hold on
% L.draw;



R.computeInclusionMesh;

[A2D,b2D]=R.computeStiffness;
Nu2D=size(A2D,1);
Nu1D=opt.uMeshN;
Nphi=opt.lambdaMeshN-1;
Npsi=opt.psiMeshN;
h=1/(Nu1D-1);
b=[b2D];
A=blkdiag(A2D);
for j=1:n
   A1D=mu(j)*spdiags([-ones(Nu1D,1)   2*ones(Nu1D,1)   -ones(Nu1D,1)]/h,[-1   0
1],Nu1D-2,Nu1D-2);
   A1D=blkdiag(A1D,speye(2));
   b1D=zeros(Nu1D,1);
```

```
    A=blkdiag(A,A1D);
    b=[b;b1D];
end
B2D=R.computeB2D;
G2D=R.computeG2D;
G1D=R.computeG1D;
Gpsi=R.computeGpsi;
C1D=R.computeC1D;
C2D=R.computeC2D;
B1D=R.computeB1D;

G=blkdiag(G2D,G1D);
C=[C2D;C1D];
B=[-B2D;B1D];
ZZ=@(r,c) sparse(r,c);

Nu=Nu2D+n*Nu1D;

M=[  G          ZZ(Nu,n*Nphi)      -C              A';
   ZZ(n*Nphi,Nu) ZZ(n*Nphi,n*Nphi) ZZ(n*Nphi,n*Npsi)      B';
    -C'        ZZ(n*Npsi,n*Nphi)     2*Gpsi     ZZ(n*Npsi,Nu);
     A              B        ZZ(Nu,n*Npsi)      ZZ(Nu,Nu)];

bM=[ZZ(Nu,1);ZZ(n*Nphi,1); ZZ(n*Npsi,1); b];
sol=M\bM;

u2D=sol(1:Nu2D);
```

```
u1D=repmat(struct('sol',zeros(Nu1D,1)),n,1);
nS=Nu2D;

for k=1:n
  uu1D=sol(nS+1:nS+Nu1D);

  nS=nS+Nu1D;
u1D(k).sol(R.inclusion(k).umesh.pivot<=R.inclusion(k).umesh.ndof)=uu1D(1:R.incl
usion(k).umesh.ndof);
u1D(k).sol(R.inclusion(k).umesh.pivot>R.inclusion(k).umesh.ndof)=b1D(end-1:end);
end

R.setSolution(u2D);
figure(1)
R.trisurf;

for k=1:n
   figure(k+1)
   plot(L(k).umesh.node,u1D(k).sol,'bx-')
end
```

Raffinamento della mesh 2D

Sono state condotte sul precedente 'Test 4 inclusioni' differenti prove per verificare come si comporta il modello al raffinamento della mesh del dominio 2D. Nello specifico, sono state create 5 mesh con gradi di libertà crescenti (vedi tabella 2).

Come esempio, viene analizzato il comportamento della soluzione 1D sulla fibra con vertici (-0,7;0) (-0,7;1), al raffinare della mesh.

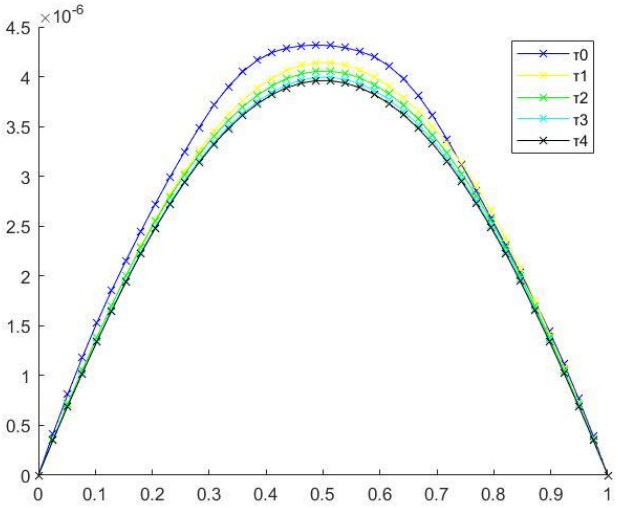


Figura 17 Soluzione per la fibra con vertici (-0,7;0) (-0,7;1) analizzata con 5 mesh con differente livello di raffinamento.

Con le triangolazioni:

Triangolazione	Ndof
τ0	143
τ1	305
τ2	1553
τ3	3107
τ4	15.500

Tabella 2 Gradi di libertà per le differenti mesh

Come si può evincere dai grafici, al raffinare della mesh diminuisce sia il punto di massimo raggiunto dalla deformazione della fibra e simultaneamente il grafico diventa sempre più simile ad una parabola, non mostrando come nel caso con mesh meno fine,

un tratto centrale nella fibra quasi a soluzione costante. Inoltre, la distanza tra le curve diminuisce, mostrando, come atteso, un cambiamento convergente al raffinamento.

Nelle seguenti figure sono rappresentate le soluzioni 2D per le triangolazioni τ0, τ1,τ3,τ4 , che risultano essere due meno fini e due più fini rispetto alla triangolazione τ2 di cui la soluzione è mostrata nel paragrafo 'test 4 inclusioni (caso ridotto 2D-1D)'.

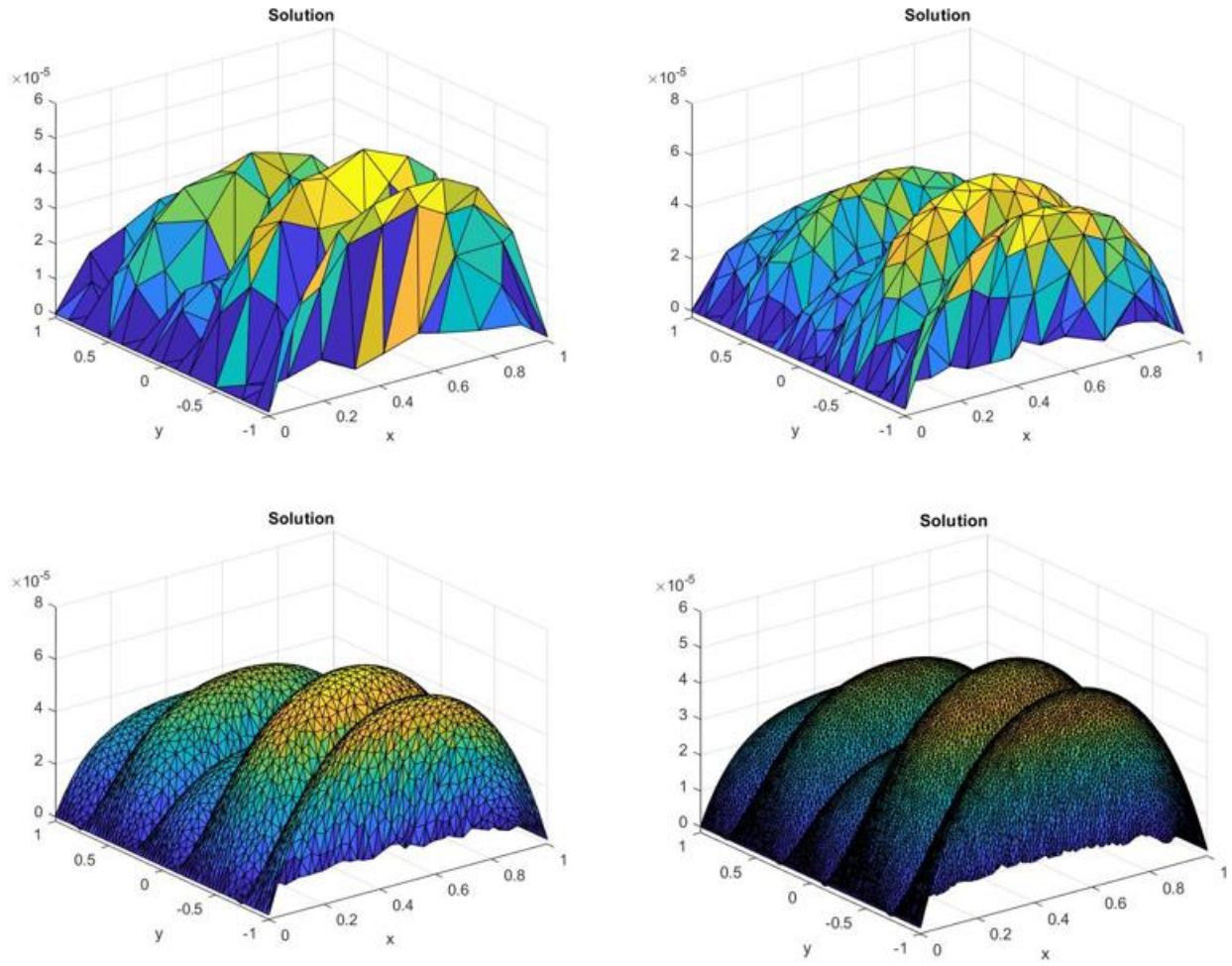


Figura 18 Soluzione per il problema accoppiato 1D-2D nel caso di vetroresina analizzata in 4 mesh con differente livello di raffinamento.

Le immagini della Figura 18 rappresentano il risultato ottenuto sulla soluzione 2D del problema accoppiato al raffinamento della mesh. La soluzione si presenta più

uniforme, meno spezzettata e in prossimità delle inclusioni sono meno evidenti punti di discontinuità nella forma di picchi e valli più o meno accentuati.

Raffinamento della mesh 1D

L'analisi che segue mostra il comportamento della soluzione al raffinamento delle mesh 1D. Si osserva che al fine di ottenere risultati più accurati è necessario raffinare simultaneamente anche la mesh 2D. In particolare, se la mesh 1D è molto più fine del diametro degli elementi 2D il sistema diventa mal condizionato.

Vengono per cui mostrati tre casi, in ordine:

1. matrice poco fine (τ0) e allo stesso modo Mesh 1D poco fine (25 elementi),
2. matrice raffinata (τ4) e allo stesso modo Mesh 1D più fine (250 elementi),
3. matrice raffinata (τ4) e Mesh 1D poco fine (25 elementi).

Il caso di matrice poco fine (τ0) e Mesh 1D più fine (250 elementi) dà origine ad un problema mal condizionato.

A seguire le immagini che rappresentano sia la soluzione del problema accoppiato sia la soluzione riguardante la fibra con vertici (-0,7;0) (-0,7;1).

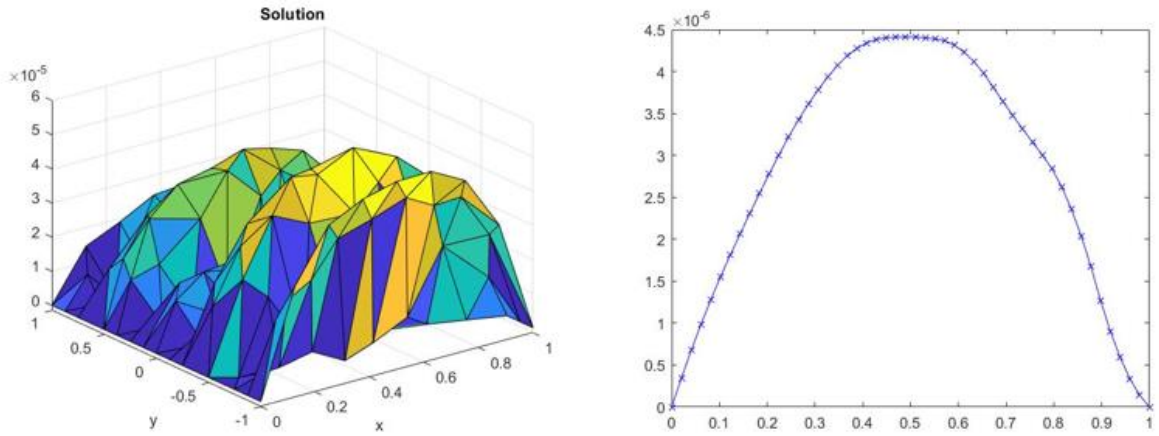


Figura 19 Matrice poco fine e allo stesso modo e psi poco fine (25 elementi)

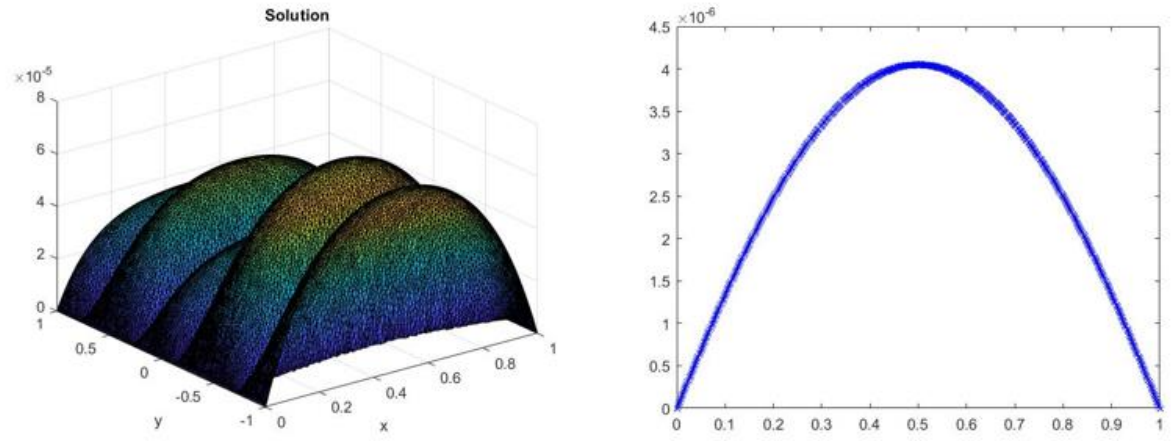


Figura 20 Matrice più fine e allo stesso modo e psi più fine (250 elementi)

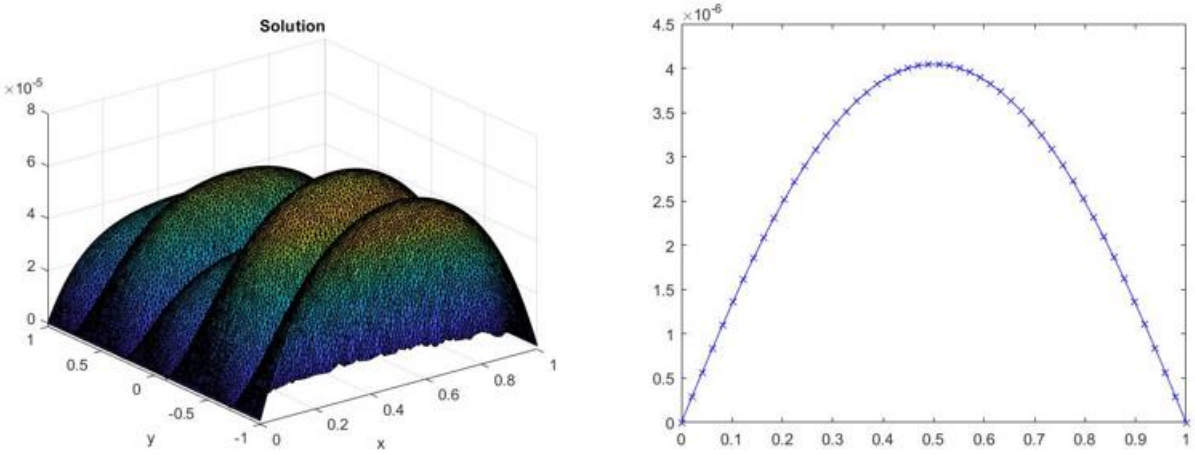


Figura 21 Matrice più fine e allo stesso modo e psi poco fine (25 elementi)

Coerentemente al modello matematico, la seconda soluzione, ovvero matrice raffinata (τ4) e allo stesso modo Mesh 1D più fine (250 elementi), risulta essere quella più precisa, continua ed uniforme. È limitata la presenza di picchi e valli in corrispondenza delle inclusioni e la soluzione riferita all'inclusione è quella più vicina ad una parabola perfetta.

Confronto delle soluzioni del caso 2D-1D con il caso 2D-2D

In questa sezione viene proposto il confronto tra le soluzioni del problema equidimensionale e del problema ridotto 2D-1D sulla fibra con vertici (-0,7;0) (-0,7;1) al variare della mesh del problema 2D. Gli andamenti sono riportati in Figura 22.

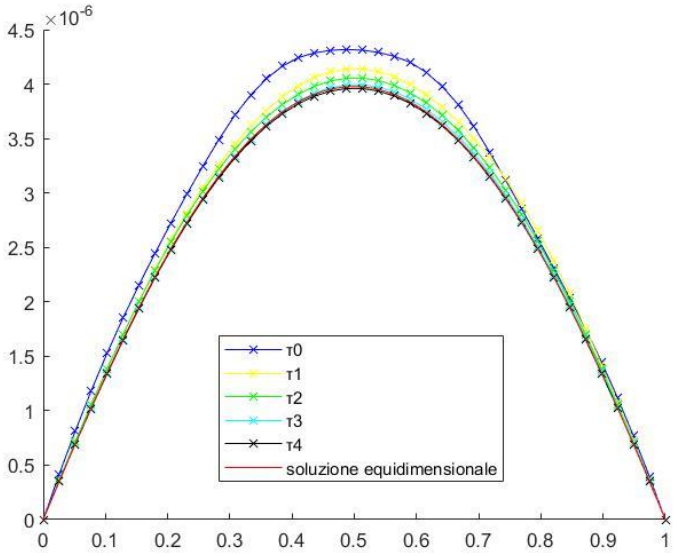


Figura 22 soluzioni fibra con vertici (-0,7;0) (-0,7;1) nel modello accoppiato 2D-1D e nel modello equidimensionale

Nel caso equidimensionale, il grafico è stato ottenuto intersecando un piano perpendicolare alla linea di mezzeria della inclusione e interpolando i punti ottenuti se ne è ricavato il grafico.

Nello specifico, è stato calcolato l'errore massimo mediante le formule anticipate nei precedenti paragrafi per le 5 differenti mesh e i risultati sono stati tabellati:

Triangolazione	Ndof	Errore
τ0	143	5,88 e-07
τ1	305	2,98 e-07
τ2	1553	1,66 e-07
τ3	3107	1,13 e-07
τ4	15.500	6,73 e-08

Tabella 3 Errore in norma infinito al variare della Mesh 2D

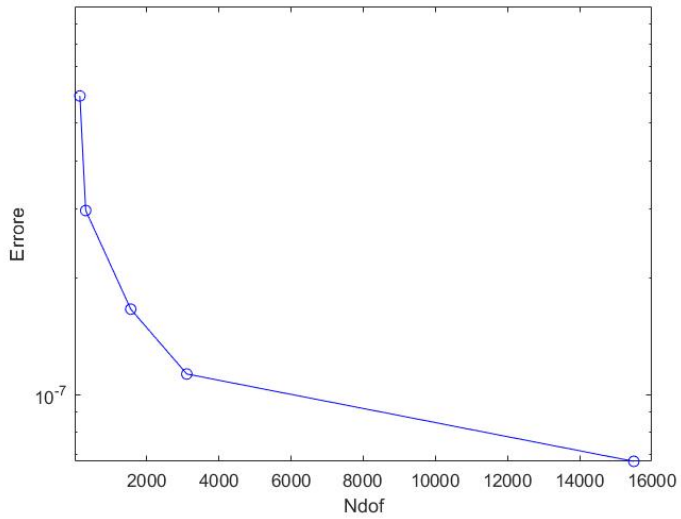


Figura 23 Andamento dell'errore al variare dei gradi di libertà (scala semi-logaritmica)

Si osserva, anche in questo caso un comportamento convergente al raffinamento della mesh.

Si mostrano in seguito anche le soluzioni 2D sia per il caso bidimensionale che per il caso equidimensionale.

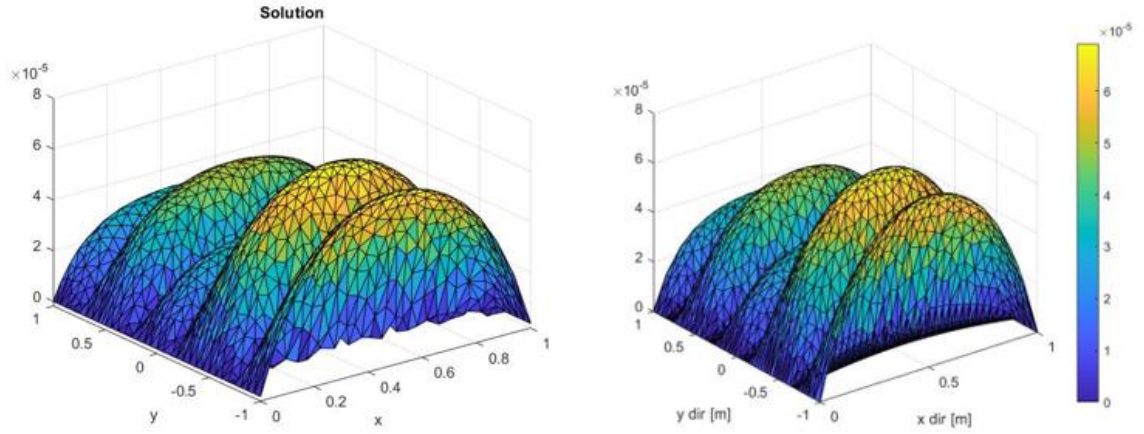


Figura 24 A sinistra la soluzione del caso bidimensionale a destra la soluzione del caso equidimensionale

Le differenze invece che si presentano in prossimità delle inclusioni in cui la soluzione e spezzettata e non lineare dipendono dalla non conformità della mesh e si riducono anch'esse al raffinamento.

Altri esempi applicativi

Per maggiore completezza nella trattazione e per dimostrare le potenzialità del codice in tale paragrafo si analizzano due casistiche alternative. La prima punta a mostrare il funzionamento del codice anche in caso di inclusioni con direzioni, lunghezze e coefficienti differenti. La seconda invece analizza un caso in cui la forzante dipende anche dalla posizione x e y e le condizioni al contorno sono differenti.

Esempio 1

Il primo esempio è mostrato per verificare e dimostrare l'applicabilità del codice nel caso di 3 inclusioni che hanno tutte direzioni, lunghezze, posizioni di partenza e coefficienti di taglio differenti.

A seguire la parte di codice aggiornata per quanto riguarda le inclusioni e la forzante.

```
L(1)=LineSeg([0; 0],[1; 0],0.8,opt);
L(2)=LineSeg([1; -0.3],[-1; -0.5],0.7,opt);
L(3)=LineSeg([0.5; 1],[-0.3; -1],0.6,opt);
mu=[100000;1000;1000000];
```

La soluzione 2D e le soluzioni delle diverse inclusioni sono le seguenti.

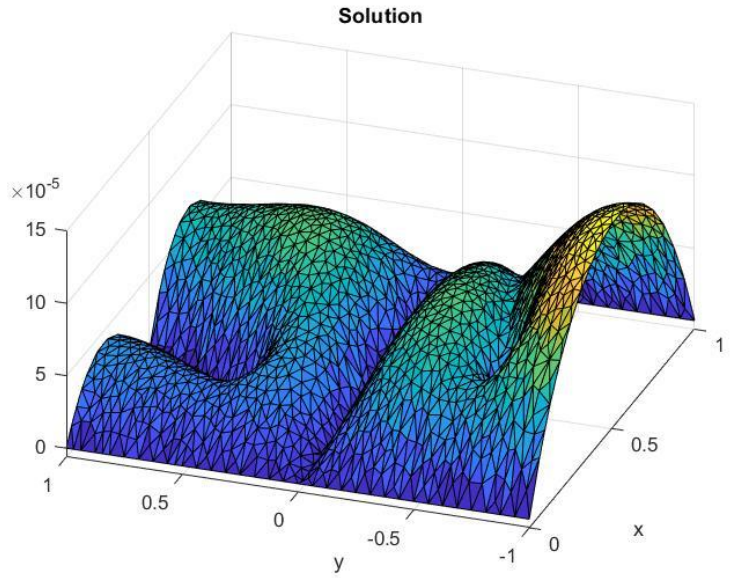


Figura 25 Soluzione 2D esempio applicativo di 3 fibre con direzioni, punti di partenza e caratteristiche differenti

Come si può evincere dalla soluzione 2D e dai grafici sotto delle soluzioni 1D, le inclusioni avendo coefficienti di taglio di differenti ordini di grandezza, avranno le rispettive deformazioni differenti, e ne conseguirà una flessione della matrice differente.

È da notare che le soluzioni delle inclusioni variano di ordini di grandezza pari a quelli dei rispettivi coefficienti di taglio e che la soluzione dell'inclusione L2 non risulta essere una parabola perfetta ma è inclinata verso la fine dell'asse x (lunghezza del segmento), dovuto al contributo derivante dalla presenza dell'inclusione L3 che è quella con coefficiente di taglio più elevato.

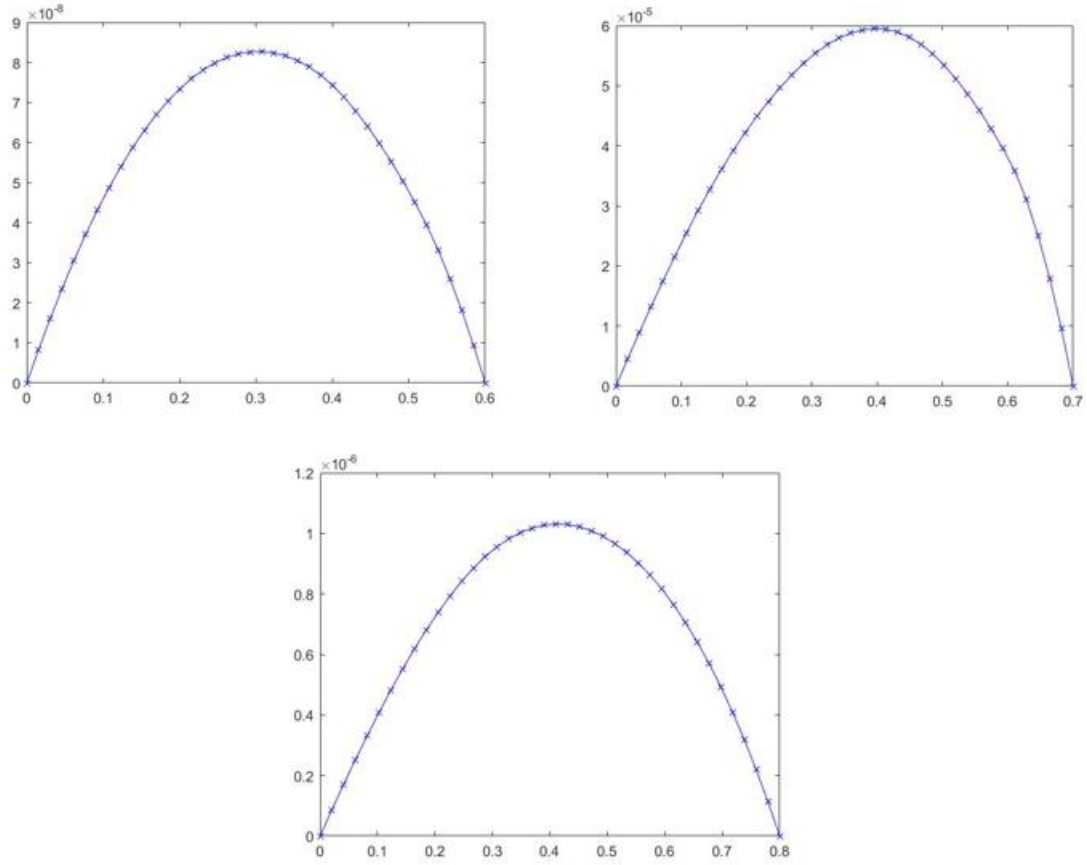


Figura 26 Soluzioni 1D esempio applicativo di 3 fibre con direzioni, punti di partenza e caratteristiche differenti

Esempio 2

Tale esempio è riconducibile ad un caso reale in cui la forzante sia proporzionale alla posizione x e y e le condizioni al contorno non siano di tipo omogeneo. A seguire la parte di codice aggiornata per quanto riguarda la forzante e le condizioni al contorno.

```
Data=ProblemData;
Data.setForcing(@(x,y) 2+300*x+400*y);
Data.setDiffusivity(1123); % E=3100 v=0.38 POLIESTERE (RESINA
TERMOIDURENTE)
Data.setGamma(0);

BC=BoundaryConditions;
BC.setBoundaryConditions(R.Points(1:4),'Dirichlet', @(x,y)
0.005*x+0.001*y);
BC.setBoundaryConditions(R.Edges([2 4]),'Dirichlet',@(x,y)
0.005*x+0.001*y);
BC.setBoundaryConditions(R.Edges(1),'Neumann',@(x,y) 10*x);
BC.setBoundaryConditions(R.Edges(3),'Neumann',1);
```

Il plot della soluzione al problema è il seguente.

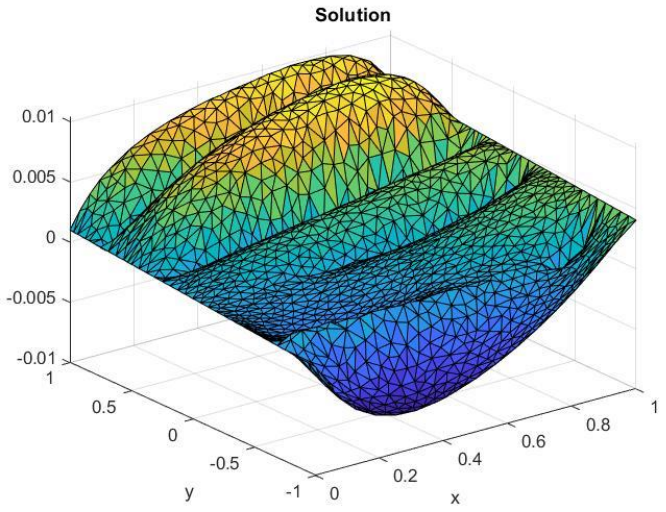


Figura 27 Soluzione 2D esempio applicativo rete di fibre perpendicolari tra loro con forzante proporzionale a x e y

Il risultato ottenuto dal codice è coerente con le aspettative poichè la soluzione cresce al crescere di x e y.

Tale risultato si riscontra anche nelle soluzioni 1D , in particolare si nota che la soluzione mantiene lo stesso ordine di grandezza ma cresce da y=-1 a y=1 e da x=0 a x=1.

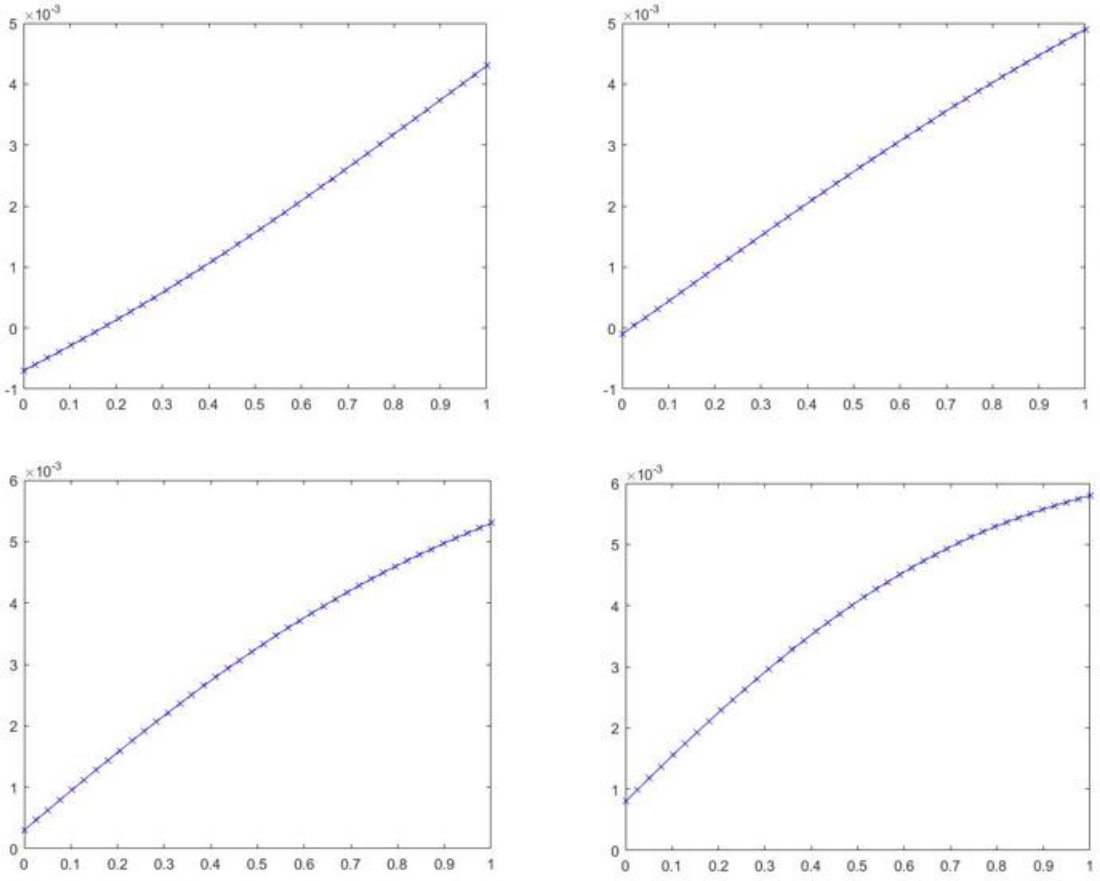


Figura 28 Soluzioni 1D esempio applicativo rete di fibre perpendicolari tra loro con forzante proporzionale a x e y

CONCLUSIONI

Lo scopo della presente tesi è stato quello di implementare un approccio numerico per la simulazione di equazioni ellittiche accoppiate bidimensionali e unidimensionali, con il fine di ridurne il costo computazionale, usando una riduzione del problema da 2D-2D a 2D-1D ,e evitando così di generare mesh all'interno delle inclusioni.

Tale obiettivo è stato raggiunto mediante una formulazione del problema basata sulla minimizzazione di un funzionale appositamente progettato per imporre le condizioni di accoppiamento alle interfacce tra i problemi 2D e 1D.

Al fine di affrontare il suddetto problema sono state di fondamentale importanza conoscenze riguardo in primis la deformazione elastica per il caso del filo elastico e della membrana elastica e poi le basi del FEM (Finite Element Method), le quali sono state approfondite e applicate al problema in questione. La formulazione variazionale discreta è stata implementata nel modello matematico centrale allo studio in questione.

È stato così possibile, partendo da un modello semplice equidimensionale, che comprende una fibra che divide il dominio 2D in due parti, ottenere il modello ridotto con tutte le matrici e vettori che ne conseguono.

La sua implementazione su ambiente Matlab ha dato i risultati che ci si aspettava. Innanzitutto, è stata riscontrata una corrispondenza tra i risultati della soluzione equidimensionale e la soluzione bidimensionale, in un caso di studio che prendeva in considerazione il materiale Vetro Resina, costituito da una matrice di poliestere e 4 fibre di vetro parallele tra loro.

Successivamente, è stato verificato che tali risultati migliorassero sia al raffinare della Mesh che al raffinare delle variabili 1D. Inoltre, al fine di ottenere risultati più accurati, se si raffina la mesh 1D è necessario raffinare simultaneamente anche la mesh 2D. In

particolare, se la mesh 1D è molto più fine del diametro degli elementi 2D il sistema diventa mal condizionato.

I risultati ottenuti hanno mostrato errori che si riducono al raffinamento della mesh.

Sono stati infine condotti altri test che hanno dimostrato la validità sia del modello matematico che del codice Matlab, analizzando casi sia con condizioni al contorno non omogenee, forzanti dipendenti dalle coordinate x e y, e fibre non parallele tra loro, inclinate e con parametri caratteristici differenti.

Il modello studiato è stato applicato a equazioni di deformazione elastica, tuttavia altri studi paralleli a codesta tesi, hanno implementato tale modello anche alla legge di Darcy e problemi di diffusione. Questo aspetto sottolinea sia la versatilità del modello matematico e allo stesso modo anche la rilevanza e solidità del modello.

Lo studio presenta ancora margini di lavoro, implementazioni e miglioramenti da un punto di vista matematico e per quanto riguarda il codice Matlab.

Può essere approfondito il caso in cui due o più inclusioni siano intersecanti tra loro. Per quanto la continuità sia garantita dal modello matematico già studiato, non è stata implementata in questo studio l'interazione tra le inclusioni nel punto di intersezione.

Altro aspetto di studio potenzialmente ancora da esplorare è quello in cui la forzante non sia perpendicolare al piano della matrice e delle inclusioni, ma che sia magari parallela o addirittura abbia direzioni ancora più particolari.

Il problema risulta affascinante e dalle variegate potenziali applicazioni di interesse ingegneristico; perciò, l'ambizione è che questa tesi e i documenti paralleli risultino importanti per gli studi e i progressi futuri.

APPENDICE A: LE CLASSI CHE COMPONGONO IL CODICE IMPLEMENTATO SU MATLAB

LE CONDIZIONI AL CONTORNO NEL PROBLEMA BIDIMENSIONALE

La classe BoundaryConditions

La classe BoundaryConditions è utilizzata per gestire le condizioni al contorno del problema in questione. Questa classe fa uso del concetto di ereditarietà in MATLAB utilizzando la classe madre matlab.mixin.Copyable, permettendo così di creare copie profonde degli oggetti. Di seguito, analizzerò la struttura e il funzionamento di questa classe.

Proprietà

- markerD e markerN: Sono proprietà private che tengono traccia degli indici usati per marcare rispettivamente le condizioni al contorno di Dirichlet e Neumann. markerD inizia da 1 e markerN da 4, è una semplice convenzione per distinguere i due tipi di condizioni al contorno. Si noti che il marker 2 è riservato per Neumann omogeneo.
- values: È una proprietà con accesso in lettura privata che memorizza gli oggetti BCvalue, una classe che detiene i valori specifici delle condizioni al contorno e che vedremo successivamente.

Costruttore

- Il costruttore inizializza markerD e markerN con i valori iniziali sopra citati e crea due oggetti BCvalue, uno senza parametri e uno con un valore di 0. Questo setup iniziale prepara l'oggetto per accettare la definizione delle condizioni al contorno.

Metodi

- setBoundaryConditions: Questo è l'unico metodo di tale classe. Permette di impostare le condizioni al contorno (Dirichlet o Neumann) su elementi di bordo specificati (che possono essere punti, spigoli, o facce del dominio di simulazione). Il metodo modifica il marcatore dell'elemento di bordo per indicare il tipo di condizione al contorno e memorizza il valore della condizione al contorno nell'array values.
    - Per ogni elemento di bordo passato come input, il metodo determina il tipo di condizione al contorno (Dirichlet o Neumann) e assegna un marcatore univoco e il valore corrispondente.
    - I marcatori per le condizioni di Dirichlet e Neumann sono incrementati di 2 ogni volta che una condizione è impostata, mantenendo così separati e univoci i marcatori per tipi diversi di condizioni al contorno.
    - Se viene specificato un tipo di condizione al contorno non riconosciuto, viene generato un errore.

In tale classe è possibile anche aggiungere ulteriori tipi di condizioni al contorno (come Robin) o per affinare i controlli sui tipi di elementi di bordo. In questo momento tale condizione non è implementata.

In sintesi, BoundaryConditions è una classe utilizzata per gestire in modo flessibile e organizzato le condizioni al contorno, facilitando la distinzione tra diversi tipi di condizioni e i loro valori associati.

I VALORI DELLE CONDIZIONI AL CONTORNO

La classe BCvalue

La classe BCvalue è progettata per rappresentare i valori delle condizioni al contorno.

Questa classe consente di gestire sia valori costanti che funzioni di valori, permettendo una maggiore flessibilità nella definizione delle condizioni al contorno.

Proprietà

- value: Una proprietà con accesso in scrittura privato che memorizza il valore della condizione al contorno. Può essere un numero (per condizioni costanti) o una funzione (per condizioni variabili nello spazio o nel tempo).
- type: Una proprietà anch'essa con accesso in scrittura privato che indica il tipo di valore memorizzato. Se type è 0, il valore è considerato costante; se è 1, il valore è considerato una funzione.

Costruttore

- Il costruttore della classe BCvalue accetta un parametro opzionale val. Se val è fornito, il costruttore imposta value su val e determina type in base al tipo di val: se val è un handle di funzione, type viene impostato su 1; altrimenti, su 0. Questo consente di creare oggetti BCvalue sia per valori costanti che per funzioni.

Metodi

- evaluate: Questo metodo permette di valutare il valore della condizione al contorno. Se il tipo è 0 (costante), semplicemente restituisce il valore costante memorizzato in value. Se il tipo è 1 (funzione), chiama la funzione memorizzata in value con gli argomenti forniti (x, y, z) e restituisce il risultato. Il metodo è

flessibile rispetto al numero di argomenti forniti, consentendo la valutazione di funzioni di variabile singola, doppia o tripla, a seconda dei casi d'uso.

La classe BCvalue è una ottima classe per gestire le condizioni al contorno ,consentendo agli utenti di specificare condizioni sia fisse che variabili. Questa flessibilità è un importante punto a favore per future implementazioni, poiché la rende versatile e utilizzabile anche in diverse casistiche.

Per esempio, in un problema di diffusione termica, che non si discosta troppo da uno di deformazione, si potrebbe voler impostare una condizione al contorno che varia nel tempo o lungo una superficie. Utilizzando BCvalue, si può facilmente definire tale condizione come una funzione del tempo o della posizione e valutarla dinamicamente durante la simulazione.

LO STUDIO DEL DOMINIO

Questa parte punta a spiegare tutta la parte di codice che concerne lo studio del dominio 2D.

La classe Domain

Questa classe è progettata per gestire e manipolare il dominio 2D per le equazioni differenziali parziali (PDE) di studio e non solo. La classe include proprietà e metodi per gestire punti, facce ed edge in un dominio, gestire vincoli di mesh e interagire con strumenti di generazione e visualizzazione della mesh. Analizziamo gli aspetti principali:

Proprietà:

- Proprietà Private: Includono identificatori, dimensionalità, percorsi delle cartelle dei dati e contatori per punti, facce ed edge. Include anche array per punti, facce, edge, un centroide (xG), punti della mesh vincolati e regioni della mesh.
- Proprietà Dipendenti: Forniscono flussi e marcatori per punti, edge e facce, facilitando l'interazione con strumenti di generazione e visualizzazione della mesh.

Metodi:

- Costruttore (Dominio): Inizializza il dominio, permettendo la specificazione della dimensionalità (2D o 3D), dei punti e l'impostazione della cartella dati.
- Gestione della Geometria: Metodi come AddPoint, AddEdge, AddFace e AddMeshConstraintPoint permettono di costruire il dominio geometrico aggiungendo singolarmente componenti.
- Inizializzazione e Computazione: Metodi come Initialize, computexG (per calcolare il centroide), computeFace e computeEdges preparano il dominio per la generazione della mesh impostando le necessarie relazioni geometriche.

- Gestione della Mesh: setMeshRegions e computeMesh sono usati per definire le regioni della mesh e generare la mesh, rispettivamente.
- Visualizzazione: draw, drawSol e plot sono metodi per visualizzare il dominio e la sua mesh usando routine di plotting personalizzate.
- Metodi di Utilità: Includono setter come setId e getter per proprietà dipendenti, fornendo un accesso semplificato ai dati geometrici e della mesh del dominio.

La classe utilizza estensivamente altre classi personalizzate (DomainPoint, DomainFace, DomainEdge), creando così un codice dove ogni entità geometrica è gestita separatamente.

La classe Domain2D

La classe denominata Domain2D, è una sottoclasse della classe più generica Domain. La classe Domain2D è progettata per gestire domini bidimensionali, con particolare attenzione alla manipolazione di inclusioni, la definizione delle condizioni al contorno, la gestione dei dati associati al dominio, la soluzione di problemi definiti sul dominio, e l'ottimizzazione di parametri specifici.

Proprietà

- inclusion: Array di oggetti LineSeg, rappresentanti segmenti di linea del dominio.
- nI: Contatore delle inclusioni presenti nel dominio.
- BC: Proprietà per memorizzare le condizioni al contorno.
- Data: Generico contenitore di dati associati al dominio.
- sol: Soluzione di un problema definito sul dominio.
- opt: Opzioni o parametri di ottimizzazione.
- Npsi, Nlambda, Nu: Dimensioni delle tre variabili psi, lambda e u.

Metodi

- addInclusion(obj, L): Aggiunge un'inculsione al dominio.
- setBC(obj, BC): Imposta le condizioni al contorno.
- setData(obj, data): Associa dati al dominio.
- setSolution(obj, sF): Definisce la soluzione del problema sul dominio.
- setOpt(obj, o): Imposta opzioni o parametri di ottimizzazione.
- computeMesh(obj, A): Inizializza o calcola la mesh per il dominio, con un approccio che sembra prevedere il caricamento da un file esterno.
- computeInclusionMesh(obj): Calcola le mesh per le inclusioni.
- computeStiffness(obj): Calcola la matrice di rigidezza.
- computeG2D(obj), computeG1D(obj), computeGpsi(obj): Calcolano le matrici del problema sopra affrontato.
- computeC2D(obj), computeC1D(obj): Calcolano le matrici C per le inclusioni e la superfice poliestere.

In generale, questa classe è progettata per supportare l'analisi dei domini bidimensionali estrapolandone le matrici finalizzate al calcolo della matrice M, la gestendo dunque la presenza di inclusioni.

BORDI, FACCE E PUNTI DEL DOMINIO

Come anticipato il codice è stato sviluppato in modo tale da trattare tutte le entità del dominio in maniera separata e dettagliata.

La classe DomainEdge

DomainEdge è una sottoclasse di matlab.mixin.Copyable. Questa classe definisce un bordo all'interno di un dominio. La classe include proprietà per l'identificazione (Id), vertici (GlobalVertexes), punti lungo il bordo (Points) e la proprietà marker.

Il costruttore (function obj=DomainEdge(id,vertexlist,pointlist)) inizializza una nuova istanza di DomainEdge con l'ID specificato, l'elenco di vertici e l'elenco di punti. La proprietà marker è impostata su 2 per impostazione predefinita.

La classe DomainFace

DomainFace è anch'essa sottoclasse di matlab.mixin.Copyable. Questa classe rappresenta una faccia all'interno di un dominio. La classe include diverse proprietà per gestire le informazioni geometriche della faccia, nonché un metodo costruttore e un metodo per impostare i bordi della faccia.

Proprietà:

- Id: Un identificatore per la faccia.
- Points: Un elenco di punti che compongono la faccia.
- GlobalVertexes: Un elenco di identificatoridei vertici globali della faccia.
- LocalVertexes: Un elenco di identificatori locali per i vertici della faccia, utile per il riferimento interno.
- nVertexes: Il numero di vertici della faccia.
- Edges: Un elenco di bordi che delimitano la faccia.
- marker: è inizializzato a 0.

- Normal: Il vettore normale alla faccia, calcolato a partire dai punti della faccia.

Metodi:

- Costruttore (DomainFace): Inizializza una nuova istanza di DomainFace con l'ID specificato, un elenco di punti, un elenco di vertici globali, un punto di riferimento xG per il calcolo della normale e un elenco di bordi . Calcola anche il vettore normale alla faccia e si assicura che la faccia abbia almeno tre vertici. Se la faccia è in uno spazio tridimensionale, calcola e, se necessario, inverte la normale per assicurare che punti all'esterno rispetto al punto xG fornito.
- setEdges: Un metodo per aggiungere uno o più bordi all'elenco dei bordi della faccia dopo la sua creazione.

La classe DomainFace fornisce quindi un modo strutturato per rappresentare e manipolare le facce geometriche in un dominio.

La classe DomainPoint

Questa classe è pensata per rappresentare un punto all'interno di un dominio. Attraverso le proprietà quali identificatori unici dei punti, coordinate e marker, il metodo costruttore (DomainPoint) inizializza un'istanza di DomainPoint con un identificatore e le coordinate fornite. Se non vengono forniti argomenti, il costruttore non eseguirà l'assegnazione delle proprietà. Il marcatore pubblico è inizializzato a 0 per default.

Tale classe offre un modo semplice ma flessibile per gestire punti .

LA MESH INDOTTA

La classe InducedMesh

Questa classe è progettata per rappresentare una mesh indotta. La classe gestisce intervalli, nodi, e valori associati ai punti e agli elementi della mesh, fornendo funzionalità per l'aggiunta di intervalli, la rimozione di duplicati, e la visualizzazione.

La classe InducedMesh è focalizzata sulla gestione di mesh indotte lungo una traiettoria o un percorso definito .La possibilità di aggiungere intervalli, insieme alla rimozione intelligente dei duplicati, facilita la creazione e la manipolazione di mesh complesse. Inoltre, il metodo di disegno integrato permette una rapida visualizzazione della mesh per scopi di verifica o analisi.

LE INCLUSIONI

La classe LineSeg

Tale codice è implementato per la gestione di segmenti di linea. Questa classe, LineSeg, è progettata per operazioni come la generazione di mesh, la proiezione, l'intersezione con elementi e la valutazione di alcune funzioni (Psi, Lambda, U) sul segmento di linea. Le operazioni coinvolgono calcoli matematici relativi alla geometria, come trovare proiezioni su linee, valutare se i punti sono su una linea e intersecare il segmento di linea con elementi poligonali di una mesh.

La classe utilizza diverse strutture dati personalizzate (InducedMesh, mesh1d, UnionMesh) per rappresentare le mesh e svolge varie computazioni geometriche. Sono inclusi compiti come il mapping tra diverse rappresentazioni di mesh, la gestione delle condizioni al contorno e la valutazione di funzioni definite a tratti attraverso le mesh.

È importante notare alcuni aspetti chiave e implicazioni della progettazione:

- Ereditarietà da matlab.mixin.Copyable: Ciò consente di fare copie profonde degli oggetti LineSeg.
- Proprietà e Metodi: Le proprietà della classe includono descrittori geometrici del segmento di linea (x0, x1, t, n, L), valori di tolleranza (tol), rappresentazioni di mesh per diverse variabili (Imesh, psimesh, lambdamesh, umesh) e opzioni (opt). I metodi forniscono funzionalità per impostare queste proprietà, eseguire calcoli geometrici e interagire con le mesh.
- Operazioni di Mesh: I metodi della classe relativi alla gestione della mesh (setUMesh, setLambdaMesh, setPsiMesh, computeInducedMesh, ecc.) sono fondamentali per la discretizzazione del dominio, la quale influenza l'accuratezza della soluzione e la convergenza.

- Calcoli Geometrici: Funzioni come ProjectionOnLine, isOnLine, IntersectWithPoly sono prettamente geometriche e servono ad identificare proiezioni, intersezioni e sovrapposizioni.

LA MESH PER LA FIBRA

La classe Mesh1d

La classe mesh1d è progettata per gestire dati di mesh unidimensionali. La classe include una varietà di proprietà e metodi per manipolare i dati della mesh. Analizziamo i componenti di questa classe:

Proprietà

- intervals: Una matrice per memorizzare gli intervalli tra i nodi nella mesh.
- elelist: Un vettore per memorizzare le liste degli elementi.
- nI: Uno scalare che indica il numero di intervalli.
- nP: Uno scalare che indica il numero di punti/nodi.
- pointlist: Una matrice per memorizzare le coordinate dei punti.
- node: Un vettore o matrice per memorizzare le informazioni dei nodi;
- bfvalues: Una matrice per memorizzare i valori delle funzioni di base.
- bfdofs: Una matrice per memorizzare i gradi di libertà associati alle funzioni di base.
- ndof: Uno scalare che indica il numero di gradi di libertà.
- order: Uno scalare che indica l'ordine della mesh (ad es., lineare, quadratico).
- pivot: Un vettore per memorizzare le informazioni di pivot per i gradi di libertà.

Metodi

- Costruttore (mesh1d): Inizializza un oggetto mesh1d, impostando opzionalmente l'ordine se fornito.

- setOrder: Metodo per impostare l'ordine della mesh.
- copyInduced: Copia i dati della mesh dagli argomenti forniti nell'oggetto, aggiustando i gradi di libertà e i valori delle funzioni di base a seconda dell'ordine e applicando opzionalmente le condizioni al contorno.
- equallySpacedMesh: Genera una mesh equidistante basata sui parametri forniti (numero di nodi, lunghezza, punto di partenza, direzione, e opzionalmente condizioni al contorno).
- stabMesh: Inizializza la mesh basata sui nodi e punti forniti direttamente, con una struttura più semplice non dipendente dalle condizioni al contorno.
- draw: Un metodo di visualizzazione per tracciare i punti della mesh.

Utilizzo

La classe è progettata per consentire la creazione e la manipolazione facile di mesh 1D, inclusa l'impostazione dei parametri della mesh, la copia dei dati della mesh, la generazione di tipi specifici di mesh (equidistanti, stabilizzate) e il tracciamento della mesh a scopi di visualizzazione.. La capacità di applicare condizioni al contorno durante la generazione o la copia della mesh ne migliora la sua applicazione nella risoluzione di problemi ai valori al contorno.

I DATI DEL PROBLEMA

La classe ProblemData

ProblemData è un tipo di oggetto progettato per immagazzinare e gestire i dati relativi a un problema. La classe eredita da matlab.mixin.Copyable.

Le proprietà sono impostate come private per l'accesso in scrittura, il che significa che possono essere modificate solo attraverso i metodi definiti nella classe e non direttamente dall'esterno. Le proprietà predefinite sono:

- diffusivity: rappresenta il coefficiente di diffusività, impostato a 1.
- velocity: un vettore per la velocità, preimpostato a [0 0].
- gamma: un coefficiente usato per rappresentare un tasso di reazione ,inizializzato a 0.
- forcing: un termine di forzamento, anch'esso inizializzato a 0.

Il costruttore ProblemData permette la creazione di un array di oggetti ProblemData con proprietà personalizzate passate come argomenti variabili. Questo meccanismo consente di inizializzare una serie di oggetti con diverse proprietà in un singolo comando.

Metodi per Impostare i Dati

I metodi setDiffusivity, setVelocity, setGamma, e setForcing permettono di modificare le proprietà dell'oggetto dopo la sua creazione. Il metodo setData offre un'interfaccia unificata per chiamare questi metodi specifici basandosi sul campo desiderato.

Metodo di Valutazione

Il metodo evaluate è particolarmente interessante perché consente di calcolare i valori delle varie proprietà (diffusività, gamma, velocità, forzamento) in base alle coordinate

fornite dalla mesh. Questo metodo assume che le proprietà possano essere sia scalari costanti che funzioni delle coordinate spaziali. Se una proprietà è una funzione, viene valutata utilizzando le coordinate cellG della mesh, che rappresentano le coordinate dei centri delle celle.

La flessibilità nel definire le proprietà come scalari o funzioni aumenta significativamente l'applicabilità della classe a vari tipi di problemi.

Bibliografia

Bardella, P. (2021). MeshToolbox_2021.

Berrone, S., Grappein, D., & Scialò, S. (2021). 3D-1D coupling on non conforming meshes via a three-field. *Journal of Computational Physics*.

Canuto, C. (2020). Appunti per il corso di MODELLI e METODI NUMERICI.

MATHWORKS. (2023). Tratto da https://it.mathworks.com/help/matlab/object-oriented-programming.html

S.Berrone, C. (2018). Unsteady advection-diffusion simulations in complex Discrete Fracture Networks with an optimization approach. *Journal of Hydrology*.

Stefano Berrone, S. P. (2017). Flow simulations in porous media with immersed intersecting fractures. *Journal of Computational Physics*.

Vincent Martn, J. J. (Aprile 19,2005). MODELING FRACTURES AND BARRIERS AS INTERFACES FOR FLOWA IN POROUS MEDIA. *SIAM*.

S. Berrone, A. D'Auria, S. Scialò, An optimization approach for flow simulations in poro-fractured media with complex geometries, Comput. Geosci. 25 (2021) 897–910, https://doi.org/10.1007/s10596-020-10029-8.

S. Berrone, S. Pieraccini, S. Scialò, On simulations of discrete fracture network flows with an optimization-based extended finite element method, SIAM J. Sci. Comput. 35 (2) (2013) A908–A935, http://dx.doi.org/10.1137/120882883

S. Berrone, D. Grappein, S. Pieraccini, S. Scialò, A three-field based optimization formulation for flow simulations in networks of fractures on nonconforming meshes, SIAM J. Sci. Comput. 43 (2) (2021) B381–B404, https://doi.org/10.1137/20M1319188.

C. Alboin, J. Jaffre, J. E. Roberts, X. Wang, and C. Serres ´ , Domain decomposition for some transmission problems in flow in porous media, in Numerical Treatment of Multiphase Flows in Porous Media (Beijing, 1999), Lecture Notes in Phys. 552, Springer-Verlag, Berlin, 2000, pp. 22–34