Elementi Finiti - Esercitazione 2
Mesh e quadratura numerica

Prof. Giancarlo Sangalli   Ivan Bioli

14 Marzo 2025

1 Meshing per Elementi Finiti

Riprendiamo come primo esercizio l'ultimo della precedente esercitazione. Come già anticipato, gli elementi finiti, invece, si basano su mesh; in questo corso, nello specifico, su *triangolazioni*, cioè mesh costituite da elementi triangolari. Ma come si descrive una triangolazione? La struttura dati essenziale per rappresentarla è formata dalle coordinate dei vertici e dalla matrice di incidenza.

Per la descrizione di una triangolazione, utilizzeremo due matrici:

- `p`: Una matrice di dimensioni $2 \times N_{\text{points}}$, dove ogni colonna rappresenta le coordinate di un vertice. In particolare, la prima riga contiene la coordinata x di ciascun punto, mentre la seconda riga contiene la coordinata y. Qui, N_{points} è il numero di nodi della mesh.
- `T`: Una matrice di dimensioni $3 \times N_{\text{tri}}$, dove ogni colonna rappresenta un triangolo e ogni riga indica l'indice di uno dei tre vertici che compongono il triangolo. Qui, N_{tri} è il numero totale di triangoli nella mesh. Gli elementi della matrice sono interi che indicano gli indici dei vertici corrispondenti nella matrice delle coordinate.

La matrice `T` descrive la connettività della mesh, ossia come i punti sono connessi per formare i triangoli, mentre la matrice `p` memorizza le coordinate spaziali di ciascun vertice.

Vi forniremo gran parte delle routines necessarie per il meshing durante il corso, utilizzando il generatore di mesh Gmsh e il pacchetto Julia `Gmsh.jl`. Le routines si trovano nel file `modules/Meshing.jl` e verranno aggiornate man mano che il corso progredisce.

Esercizio 1

L'obiettivo di questo esercizio è familiarizzare con la struttura dati di una triangolazione. Ecco le prime linee di codice che dovreste aggiungere al vostro script Julia:

```julia
using Revise
includet("<PATH-TO-FOLDER>/Meshing.jl")
```

- `using Revise`: Questo comando carica il pacchetto `Revise.jl`, che consente di ricaricare automaticamente i file modificati durante la sessione, senza bisogno di riavviare Julia.

- `includet("<PATH-TO-FOLDER>/Meshing.jl")`: Questa riga include il modulo `Meshing.jl`, che contiene tutte le funzioni necessarie per lavorare con le mesh. Sostituite `<PATH-TO-FOLDER>` con il percorso corretto del file `Meshing.jl`.

Una volta che queste righe sono state aggiunte, potrete iniziare a sperimentare con le mesh, utilizzando anche la documentazione delle funzioni nel file `Meshing.jl`.

1. Usando le funzioni `mesh_square`, `mesh_circle` e `get_nodes_connectivity`, eseguire alcune mesh per il quadrato e il cerchio unitario con diverse dimensioni della mesh h.

2. Definire una funzione `plot_mesh` che prende in input la matrice di incidenza `T`, le coordinate dei vertici `p` e disegna la triangolazione. La funzione deve iterare su ciascun triangolo e disegnare i bordi uno per uno. Non è necessario concentrarsi sull'efficienza in questa fase, l'obiettivo è essere sicuri di aver compreso correttamente la struttura dei dati.

3. Confrontare il risultato ottenuto con la vostra funzione di visualizzazione con quello ottenuto tramite il pacchetto `Meshes.jl` utilizzando il seguente codice:

```julia
import Meshes
mesh = to_Meshes(T, p)
Meshes.viz(mesh, showsegments = true)
```

2 Quadratura numerica

Data una triangolazione $\mathcal{T}_h$, per ogni triangolo $T \in \mathcal{T}_h$ denotiamo con $\{\mathbf{v}_T^1, \mathbf{v}_T^2, \mathbf{v}_T^3\}$ i suoi vertici e con

$$\mathbf{b}_T := \frac{\mathbf{v}_T^1 + \mathbf{v}_T^2 + \mathbf{v}_T^3}{3}$$

il suo baricentro. Indichiamo con $\mathbb{P}_0^{\text{disc}}(\mathcal{T}_h)$ lo spazio delle funzioni costanti a tratti e con $\mathbb{P}_1(\mathcal{T}_h)$ lo spazio delle funzioni lineari a tratti e continue. Si noti che una funzione in $\mathbb{P}_0^{\text{disc}}(\mathcal{T}_h)$ è individuata dai valori che assume nei baricentri (corrispondenza biunivoca), mentre una funzione in $\mathbb{P}_1(\mathcal{T}_h)$ è individuata dai valori che assume nei vertici di $\mathcal{T}_h$. Data una funzione $u : \Omega \to \mathbb{R}$, possiamo quindi definire le sue interpolazioni nei suddetti spazi:

- **Interpolazione costante a tratti** $\Pi_0 u \in \mathbb{P}_0^{\text{disc}}(\mathcal{T}_h)$, imponendo
$$(\Pi_0 u)(\mathbf{b}_T) = u(\mathbf{b}_T), \quad \forall T \in \mathcal{T}_h.$$

- **Interpolazione lineare a tratti** $\Pi_1 u \in \mathbb{P}_1(\mathcal{T}_h)$, imponendo
$$(\Pi_1 u)(\mathbf{v}_T^i) = u(\mathbf{v}_T^i), \quad \forall T \in \mathcal{T}_h, \forall i = 1, 2, 3.$$

Questo ci permette di definire le seguenti formule di quadratura per approssimare $\int_\Omega u(\mathbf{x})\,\mathrm{d}\mathbf{x}$:

$$Q_0(u) := \int_\Omega (\Pi_0 u)(\mathbf{x})\,\mathrm{d}\mathbf{x} = \sum_{T \in \mathcal{T}_h} \int_T (\Pi_0 u)(\mathbf{x})\,\mathrm{d}\mathbf{x} = \sum_{T \in \mathcal{T}_h} |T| u(\mathbf{b}_T) \tag{1}$$

$$Q_1(u) := \int_\Omega (\Pi_1 u)(\mathbf{x})\,\mathrm{d}\mathbf{x} = \sum_{T \in \mathcal{T}_h} \int_T (\Pi_1 u)(\mathbf{x})\,\mathrm{d}\mathbf{x} = \sum_{T \in \mathcal{T}_h} |T| \frac{u(\mathbf{v}_T^1) + u(\mathbf{v}_T^2) + u(\mathbf{v}_T^3)}{3} \tag{2}$$

Le precedenti formule di quadratura sono entrambe esatte se u è lineare a tratti. Una formula di quadratura più accurata, esatta sulle funzioni quadratiche, è la seguente

$$Q_2(u) := \sum_{T \in \mathcal{T}_h} |T| \frac{u(\mathbf{m}_T^1) + u(\mathbf{m}_T^2) + u(\mathbf{m}_T^3)}{3}, \tag{3}$$

dove $\{\mathbf{m}_T^1, \mathbf{m}_T^2, \mathbf{m}_T^3\}$ sono i punti medi dei lati di ogni triangolo.

Esercizio 2

Dimostrare che Q_0 e Q_1 sono esatte se u è lineare a tratti.

Esercizio 3

Misurare numericamente l'ordine di convergenza delle precedenti formule di quadratura, cioè:

1. Scegliere una funzione $u : \Omega \to \mathbb{R}$ definita su un dominio Ω in modo che $\int_\Omega u(\mathbf{x})\,\mathrm{d}\mathbf{x}$ sia calcolabile analiticamente.
2. Calcolare l'errore di quadratura
$$e_i(u) = \left| \int_\Omega u(\mathbf{x})\,\mathrm{d}\mathbf{x} - Q_i(u) \right|$$
per $i = 1, 2, 3$ e diverse $\mathcal{T}_h$ (con diversa meshsize h).
3. Fare un diagramma in scala logaritmica dell'errore contro h.
4. Provare con u lineare, quadratica o non polinomiale.

Per costruire una mesh del quadrato o del cerchio si utilizzino rispettivamente le funzioni `mesh_square` e `mesh_circle`, insieme alla funzione `get_nodes_connectivity` che permette di ottenere le coordinate dei nodi e la matrice di incidenza. Tutte le funzioni sono fornite nel file `Meshing.jl`. Alcuni esempi che si possono considerare per verificare la correttezza della propria implementazione sono mostrati nella Figura 1.

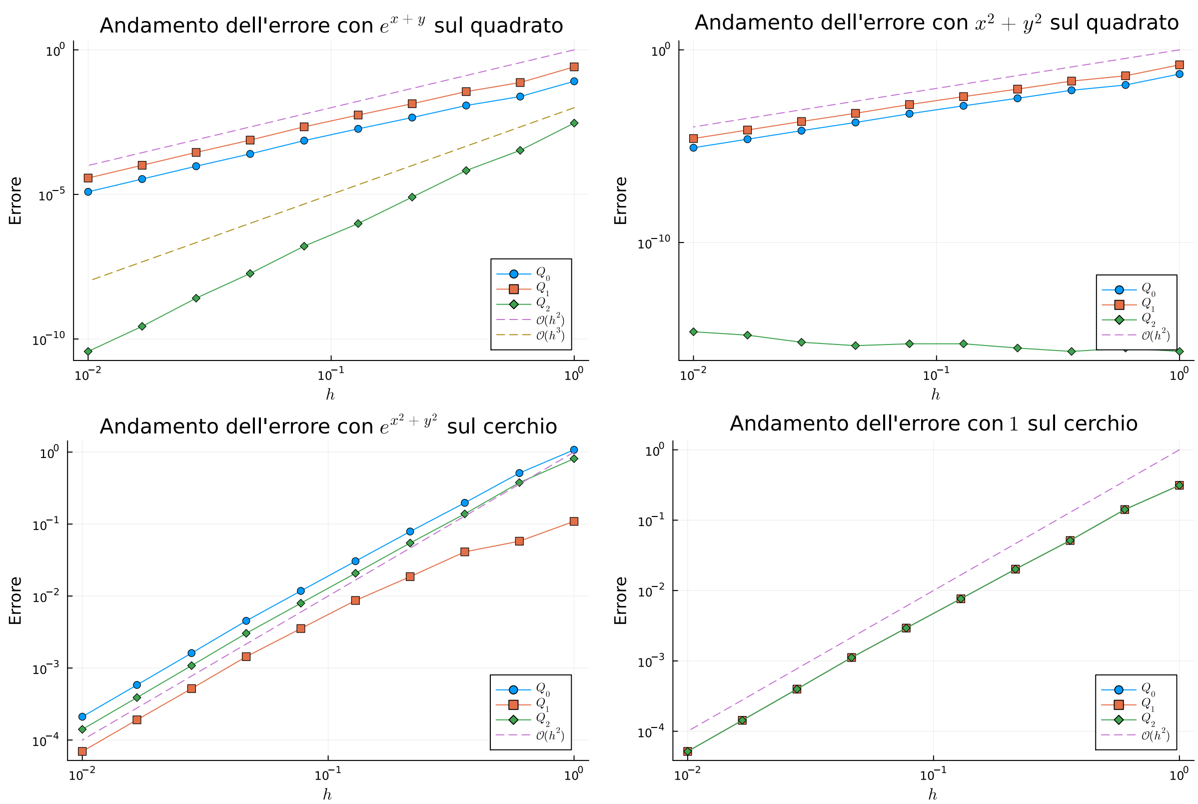

Figura 1: Alcuni esempi che si possono considerare per verificare una corretta implementazione.