Implementazione con interfaccia grafica del metodo Galerkin agli elementi spettrali con integrazione numerica su domini poligonali

Jon Matteo Church [709752]

19 settembre 2011

Indice

Capitolo 1

Introduzione

1.1 Obiettivi

Scopo del progetto è l'implementazione del metodo agli elementi spettrali con integrazione numerica, SEM-NI (Spectral Elements Method with Numerical Integration), per problemi in forma ellittica definiti su domini poligonali generici (in due dimensioni) che presentino eventualmente anche cavità.

A tale scopo vogliamo usare il linguaggio di programmazione C++ orientato agli oggetti. Tale scelta permette una maggiore facilità di lettura del codice ed una implementazione più vicina alla formulazione matematica del problema.

Vogliamo, inoltre, sviluppare una interfaccia grafica, GUI (Graphical User Interface), che permetta all'utilizzatore di definire con facilità il problema da esaminare, stablirne i parametri per la risoluzione numerica, visualizzarne la soluzione ottenuta ed, eventualmente, salvarla in file. Da ultimo vogliamo che il codice risulti portabile, ovvero, che ne sia possibile la compilazione e l'utilizzo su più sistemi operativi. Vogliamo, in particolare, che sia possibile eseguire il programma oltre che in ambiente GNU/Linux, anche su Apple Mac OS X e su Microsoft Windows.

1.2 Problema

Presentiamo di seguito la formulazione matematica generale del problema ellittico in analisi e della sua riformulazione numerica con il metodo SEM-NI.

1.2.1 Forma forte

Consideriamo equazioni della forma

$$\mathcal{L}u(\mathbf{x}) = f(\mathbf{x}), \qquad \forall \mathbf{x} \in \Omega \tag{1.1a}$$

dove Ω è un dominio poligonale di $\mathbb{R}^2$, $f : \Omega \to \mathbb{R}$ è una funzione assegnata ed $\mathcal{L}$ è un generico operatore ellittico della forma

$$\mathcal{L}u := -\nabla \cdot (\mu \nabla u) + \boldsymbol{\beta} \cdot \nabla u + \sigma u$$

essendo $\mu : \Omega \to \mathbb{R}$, $\boldsymbol{\beta} : \Omega \to \mathbb{R}^2$ e $\sigma : \Omega \to \mathbb{R}$ funzioni assegnate.

Ci limitiamo, per semplicità a considerare il caso di domini polygonali bidimensionali con cavità. Un *poligono con buchi bidimensionale* è costituito da una curva poligonale chiusa, detto *perimetro esterno*, e da un insieme, eventualmente vuoto, di curve poligonali chiuse disgiunte giacenti nella parte interna del perimetro esterno, dette *perimetri interni* o *buchi*. L'intersezione della parte interna del perimetro esterno e della parte esterna dei buchi è la parte interna del poligono con buchi.

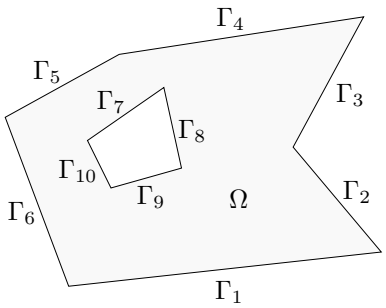

Figura 1.1: Esempio di dominio poligonale

Indichiamo con Γ_i $(i = 1, \ldots, N_l)$ i lati di Ω e completiamo l'equazione (1.1a) con le seguenti condizioni al bordo,

$$u(\mathbf{x}) = g_i(\mathbf{x}), \qquad \forall \mathbf{x} \in \Gamma_i, i \in \mathcal{I}_D \tag{1.1b}$$

$$\nabla u(\mathbf{x}) \cdot \mathbf{n}_i = h_i(\mathbf{x}), \qquad \forall \mathbf{x} \in \Gamma_i, i \in \mathcal{I}_N \tag{1.1c}$$

$$\nabla u(\mathbf{x}) \cdot \mathbf{n}_i + \gamma_i(\mathbf{x}) u(\mathbf{x}) = r_i(\mathbf{x}), \qquad \forall \mathbf{x} \in \Gamma_i, i \in \mathcal{I}_R \tag{1.1d}$$

dove $\mathcal{I}_D$, $\mathcal{I}_N$, $\mathcal{I}_R$ sono gli insiemi degli indici dei lati di Dirichlet, di Neumann e di Robin rispettivamente ($\bigcup_{i \in \mathcal{I}_D} \Gamma_i = \Gamma_D$, $\bigcup_{i \in \mathcal{I}_N} \Gamma_i = \Gamma_N$, $\bigcup_{i \in \mathcal{I}_R} \Gamma_i = \Gamma_R$, $\Gamma_D \cup \Gamma_N \cup \Gamma_R = \partial\Omega$, $\mathring{\Gamma}_D \cap \mathring{\Gamma}_N = \mathring{\Gamma}_D \cap \mathring{\Gamma}_R = \mathring{\Gamma}_N \cap \mathring{\Gamma}_R = \varnothing$) e $\mathbf{n}_i$ indica il versore normale uscente da Ω su Γ_i e $g_i : \Omega \to \mathbb{R}$, $h_i : \Omega \to \mathbb{R}$, $\gamma_i : \Omega \to \mathbb{R}$ ed $r_i : \Omega \to \mathbb{R}$ sono funzioni assegnate.

1.2.2 Forma debole

Moltiplichiamo l'equazione (1.1a) per una generica funzione test $v : \Omega \to \mathbb{R}$ ed integriamo sul dominio Ω:

$$\int_\Omega \mathcal{L}u(\mathbf{x}) v(\mathbf{x}) \mathrm{d}\Omega = \int_\Omega f(\mathbf{x}) v(\mathbf{x}) \mathrm{d}\Omega$$

$$-\int_\Omega \nabla \cdot [\mu(\mathbf{x}) \nabla u(\mathbf{x})]\, v(\mathbf{x}) \mathrm{d}\Omega + \int_\Omega \boldsymbol{\beta}(\mathbf{x}) \cdot \nabla u(\mathbf{x}) v(\mathbf{x}) \mathrm{d}\Omega + \\ + \int_\Omega \sigma u(\mathbf{x}) v(\mathbf{x}) \mathrm{d}\Omega = \int_\Omega f(\mathbf{x}) v(\mathbf{x}) \mathrm{d}\Omega,$$

applicando le formule di Green otteniamo

$$\int_\Omega \mu(\mathbf{x})\nabla u(\mathbf{x})\cdot\nabla v(\mathbf{x})\mathrm{d}\Omega - \sum_i^{N_l}\int_{\Gamma_i}\mu(\mathbf{x})\nabla u(\mathbf{x})\cdot\mathbf{n}_i v(\mathbf{x})\mathrm{d}\Gamma_i + \\ + \int_\Omega \boldsymbol{\beta}(\mathbf{x})\cdot\nabla u(\mathbf{x})v(\mathbf{x})\mathrm{d}\Omega + \int_\Omega \sigma u(\mathbf{x})v(\mathbf{x})\mathrm{d}\Omega = \int_\Omega f(\mathbf{x})v(\mathbf{x})\mathrm{d}\Omega,$$

imponendo le condizioni sui lati di Neumann e Robin otteniamo

$$\int_\Omega \mu(\mathbf{x})\nabla u(\mathbf{x})\cdot\nabla v(\mathbf{x})\mathrm{d}\Omega + \int_\Omega \boldsymbol{\beta}(\mathbf{x})\cdot\nabla u(\mathbf{x})v(\mathbf{x})\mathrm{d}\Omega + \\ + \int_\Omega \sigma u(\mathbf{x})v(\mathbf{x})\mathrm{d}\Omega - \sum_{i\in\mathcal{I}_D}\int_{\Gamma_i}\mu(\mathbf{x})\nabla u(\mathbf{x})\cdot\mathbf{n}_i v(\mathbf{x})\mathrm{d}\Gamma_i + \\ + \sum_{i\in\mathcal{I}_R}\int_{\Gamma_i}\mu(\mathbf{x})\gamma_i(\mathbf{x})u(\mathbf{x})v(\mathbf{x})\mathrm{d}\Gamma_i = \int_\Omega f(\mathbf{x})v(\mathbf{x})\mathrm{d}\Omega + \\ + \sum_{i\in\mathcal{I}_N}\int_{\Gamma_i}\mu(\mathbf{x})h_i(\mathbf{x})v(\mathbf{x})\mathrm{d}\Gamma_i + \sum_{i\in\mathcal{I}_R}\int_{\Gamma_i}\mu(\mathbf{x})r_i(\mathbf{x})v(\mathbf{x})\mathrm{d}\Gamma_i.$$

Restano da imporre le condizioni sui lati di Dirichlet che imponiamo come penalità. Sia quindi $\eta \gg 0$ in modo che sui lati di Dirichlet $\mu\nabla u\cdot\mathbf{n}_i$ sia trascurabile rispetto a ηu; otteniamo così

$$\int_\Omega \mu(\mathbf{x})\nabla u(\mathbf{x})\cdot\nabla v(\mathbf{x})\mathrm{d}\Omega + \int_\Omega \boldsymbol{\beta}(\mathbf{x})\cdot\nabla u(\mathbf{x})v(\mathbf{x})\mathrm{d}\Omega + \\ + \int_\Omega \sigma u(\mathbf{x})v(\mathbf{x})\mathrm{d}\Omega + \sum_{i\in\mathcal{I}_D}\int_{\Gamma_i}\eta u(\mathbf{x})v(\mathbf{x})\mathrm{d}\Gamma_i + \\ + \sum_{i\in\mathcal{I}_R}\int_{\Gamma_i}\mu(\mathbf{x})\gamma_i(\mathbf{x})u(\mathbf{x})v(\mathbf{x})\mathrm{d}\Gamma_i = \int_\Omega f(\mathbf{x})v(\mathbf{x})\mathrm{d}\Omega + \\ + \sum_{i\in\mathcal{I}_D}\int_{\Gamma_i}\eta g_i(\mathbf{x})v(\mathbf{x})\mathrm{d}\Gamma_i + \sum_{i\in\mathcal{I}_N}\int_{\Gamma_i}\mu(\mathbf{x})h_i(\mathbf{x})v(\mathbf{x})\mathrm{d}\Gamma_i + \\ + \sum_{i\in\mathcal{I}_R}\int_{\Gamma_i}\mu(\mathbf{x})r_i(\mathbf{x})v(\mathbf{x})\mathrm{d}\Gamma_i.$$

Affinché gli integrali a cui siamo giunti siano definiti secondo Lebesgue siano $u, v \in H^1(\Omega)$, $\mu \in L^\infty(\Omega)$, $\boldsymbol{\beta} \in [L^\infty(\Omega)]^2$, $\sigma, f \in L^2(\Omega)$ e siano i dati di bordo, γ_i, g_i, h_i, ed r_i sufficientemente regolari.

Indichiamo ora con $\langle\cdot,\cdot\rangle$ il prodotto scalare in $L^2(\Omega)$ e, senza ambiguità, anche quello su $\left[L^2(\Omega)\right]^2$:

$$\langle\phi,\psi\rangle := \int_\Omega \phi(\mathbf{x})\psi(\mathbf{x})\mathrm{d}\Omega \qquad \forall\phi,\psi\in L^2(\Omega), \tag{1.2}$$

$$\langle\boldsymbol{\phi},\boldsymbol{\psi}\rangle := \int_\Omega \boldsymbol{\phi}(\mathbf{x})\cdot\boldsymbol{\psi}(\mathbf{x})\mathrm{d}\Omega \qquad \forall\boldsymbol{\phi},\boldsymbol{\psi}\in\left[L^2(\Omega)\right]^2 \tag{1.3}$$

e con $\langle\!\langle\cdot,\cdot\rangle\!\rangle_i$ il prodotto scalare di $L^2(\Gamma_i)$

$$\langle\!\langle\phi,\psi\rangle\!\rangle_i := \int_{\Gamma_i} \phi(\mathbf{x})\psi(\mathbf{x})\mathrm{d}\Gamma_i \qquad \forall\phi,\psi\in L^2(\Gamma_i). \tag{1.4}$$

Giungiamo così alla seguente formulazione debole di (1.1):

trovare $u \in H^1(\Omega)$ tale che $\forall v \in H^1(\Omega)$

$$\langle \mu \nabla u, \nabla v \rangle + \langle \boldsymbol{\beta} \cdot \nabla u, v \rangle + \langle \sigma u, v \rangle + \eta \sum_{i \in \mathcal{I}_D} \langle\!\langle u, v \rangle\!\rangle_i + \sum_{i \in \mathcal{I}_R} \langle\!\langle \mu \gamma_i u, v \rangle\!\rangle_i$$
$$= \langle f, v \rangle + \eta \sum_{\mathcal{I}_D} \langle\!\langle g_i, v \rangle\!\rangle_i + \sum_{\mathcal{I}_N} \langle\!\langle \mu h_i, v \rangle\!\rangle_i + \sum_{\mathcal{I}_R} \langle\!\langle \mu r_i, v \rangle\!\rangle_i. \quad (1.5)$$

1.2.2.1 Buona positura

Vogliamo dimostrare la buona positura di (1.5) ricorrendo al lemma di Lax-Milgram. A tale fine vogliamo dimostrare la continuità e la coecività in $H^1(\Omega)$ della seguente forma bilineare su $H^1(\Omega)$

$$a(\phi, \psi) := \langle \mu \nabla \phi, \nabla \psi \rangle + \langle \boldsymbol{\beta} \cdot \nabla \phi, \psi \rangle + \langle \sigma \phi, \psi \rangle + \eta \sum_{i \in \mathcal{I}_D} \langle\!\langle \phi, \psi \rangle\!\rangle_i + \sum_{i \in \mathcal{I}_R} \langle\!\langle \mu \gamma_i u, v \rangle\!\rangle_i$$

e la continuità in $H^1(\Omega)$ del seguente funzionale lineare su $H^1(\Omega)$

$$F\psi := \langle f, \psi \rangle + \eta \sum_{\mathcal{I}_D} \langle\!\langle g_i, \psi \rangle\!\rangle_i + \sum_{\mathcal{I}_N} \langle\!\langle \mu h_i, \psi \rangle\!\rangle_i + \sum_{\mathcal{I}_R} \langle\!\langle \mu r_i, \psi \rangle\!\rangle_i.$$

Preliminarmente, introduciamo le seguenti notazioni per la norma su $L^p(\Omega)$

$$\|\phi\|_p = \begin{cases} \left(\int_\Omega |\phi(\mathbf{x})^p | \mathrm{d}\Omega \right)^{\frac{1}{p}} & \text{se } 1 \leq p < \infty \\ \operatorname{ess\,sup}_{\mathbf{x} \in \Omega} |\phi(\mathbf{x})| & \text{se } p = \infty \end{cases},$$

per la norma su $\left[L^2(\Omega)\right]^2$

$$|\boldsymbol{\phi}| = \left(\int_\Omega |\boldsymbol{\phi}(\mathbf{x})|^2 \mathrm{d}\Omega \right)^{\frac{1}{2}},$$

e per la norma su $L^2(\Gamma_i)$

$$|||\phi|||_i = \left(\int_{\Gamma_i} |\phi(\mathbf{x})|^2 \mathrm{d}\Gamma_i \right)^{\frac{1}{2}}$$

e ricordiamo i seguanti risultati di analisi funzionale:

Diseguaglianza di Cauchy-Schwarz siano $(V, \langle \cdot, \cdot \rangle)$ uno spazio prehilbertiano e $\| \cdot \|$ la norma indotta dal prodotto scalare su V, allora $\forall \phi, \psi \in V$

$$| \langle \phi, \psi \rangle | < \|\phi\| \|\psi\|.$$

Diseguaglianza di Hölder siano Ω uno spazio di misura e $1 \leq p, q \leq \infty$ coniugati di Hölder, ovvero tali ch e$\frac{1}{p} + \frac{1}{q} = 1$ usando la convenzione $\frac{1}{\infty} = 0$, allora se $\phi \in L^p(\Omega)$ e $\psi \in L^q(\Omega)$ si ha $\phi\psi \in L^1(\Omega)$ e vale

$$\|\phi\psi\|_1 \leq \|\phi\|_p \|\psi\|_q.$$

Diseguaglianza di traccia siano Ω un aperto limitato di $\mathbb{R}^n$, Γ una porzione regolare a misura non nulla di $\partial\Omega$ e $k \geq 1$, allora se $\phi \in H^k(\Omega) \cap C^0(\overline{\Omega})$ si ha $\phi|_\Gamma \in L^2(\Gamma)$ e vale

$$\|\phi|_\Gamma\|_{L^2(\Gamma)} \leq C_\Gamma \|\phi\|_{H^k(\Omega)}$$

per una opportuna costante positiva C_Γ.

Dimostrazione della continuità di a Per ogni ϕ, ψ in $H^1(\Omega)$ si ha:

$$\begin{aligned}
|a(\phi,\psi)| \leq & |\langle \mu\nabla\phi, \nabla\psi\rangle| + |\langle \boldsymbol{\beta}\cdot\nabla\phi, \psi\rangle| + |\langle \sigma\phi, \psi\rangle| + \eta \sum_{i\in\mathcal{I}_D} |\langle\!\langle \phi, \psi\rangle\!\rangle_i| + \\
& + \sum_{i\in\mathcal{I}_R} |\langle\!\langle \mu\gamma_i\phi, \psi\rangle\!\rangle_i| \\
\leq & \|\mu\|_\infty \|\nabla\phi\|_2 \|\nabla\psi\|_2 + \|\boldsymbol{\beta}\|_\infty \|\nabla\phi\|_2 \|\psi\|_2 + \|\sigma\|_2 \|\phi\psi\|_2 \\
& + \eta \sum_{i\in\mathcal{I}_D} |\!|\!|\phi|\!|\!|_i |\!|\!|\psi|\!|\!|_i + \|\mu\|_\infty \sum_{i\in\mathcal{I}_R} \|\gamma_i\|_{L^\infty(\Gamma_i)} |\!|\!|\phi|\!|\!|_i |\!|\!|\psi|\!|\!|_i \\
\leq & \|\mu\|_\infty \|\phi\|_{H^1(\Omega)} \|\psi\|_{H^1(\Omega)} + \|\boldsymbol{\beta}\|_\infty \|\phi\|_{H^1(\Omega)} \|\psi\|_{H^1(\Omega)} \\
& + \|\sigma\|_2 \|\phi\|_4 \|\psi\|_4 + \eta \sum_{i\in\mathcal{I}_D} C_{\Gamma_i} \|\phi\|_{H^1(\Omega)} \|\psi\|_{H^1(\Omega)} \\
& + \|\mu\|_\infty \sum_{i\in\mathcal{I}_R} \|\gamma_i\|_{L^\infty(\Gamma_i)} \|\phi\|_{H^1(\Omega)} \|\psi\|_{H^1(\Omega)} \\
\leq & \|\mu\|_\infty \|\phi\|_{H^1(\Omega)} \|\psi\|_{H^1(\Omega)} + \|\boldsymbol{\beta}\|_\infty \|\phi\|_{H^1(\Omega)} \|\psi\|_{H^1(\Omega)} \\
& + \|\sigma\|_2 \|\phi\|_{H^1(\Omega)} \|\psi\|_{H^1(\Omega)} + \eta C_D \|\phi\|_{H^1(\Omega)} \|\psi\|_{H^1(\Omega)} \\
& + \|\mu\|_\infty C_R \|\phi\|_{H^1(\Omega)} \|\psi\|_{H^1(\Omega)} \\
\leq & \left[\|\mu\|_\infty + \|\boldsymbol{\beta}\|_\infty + \|\sigma\|_2 + \eta C_D + \|\mu\|_\infty C_R\right] \|\phi\|_{H^1(\Omega)} \|\psi\|_{H^1(\Omega)} \\
\leq & C \|\phi\|_{H^1(\Omega)} \|\psi\|_{H^1(\Omega)},
\end{aligned}$$

dove abbiamo sfruttato le diseguaglianze triangolare, di Hölder, di Cauchy-Schwarz, e di traccia e la definizione di norma in $H^1(\Omega)$

$$\|\phi\|_{H^1(\Omega)} := \left(\|\phi\|_2^2 + \|\nabla\phi\|_2^2\right)^{\frac{1}{2}}.$$

La costante di continuità vale pertanto

$$C = \|\mu\|_\infty + \|\boldsymbol{\beta}\|_\infty + \|\sigma\|_2 + \eta C_D + \|\mu\|_\infty C_R,$$

avendo indicato con $C_D = \sum_{i\in\mathcal{I}_D} C_{\Gamma_i}$ e $C_R = \sum_{i\in\mathcal{I}_R} \|\gamma_i\|_{L^\infty(\Gamma_i)}$.

Dimostrazione della coercività di a Per ogni ϕ in $H^1(\Omega)$ si ha:

$$\begin{aligned}
a(\phi,\phi) =& \langle \mu\nabla\phi, \nabla\phi\rangle + \langle \boldsymbol{\beta}\cdot\nabla\phi, \phi\rangle + \langle \sigma\phi,\phi\rangle + \eta\sum_{i\in\mathcal{I}_D}\langle\!\langle \phi,\phi\rangle\!\rangle_i + \sum_{i\in\mathcal{I}_R}\langle\!\langle \mu\gamma_i\phi,\phi\rangle\!\rangle_i \\
\geq& \mu_0\|\nabla\phi\|_2^2 - |\langle \boldsymbol{\beta}\cdot\nabla\phi,\phi\rangle| + \sigma_0\|\phi\|_2^2 + \eta\sum_{i\in\mathcal{I}_D}|\!|\!|\phi|\!|\!|_i^2 + \mu_0\sum_{i\in\mathcal{I}_R}\gamma_{i,0}|\!|\!|\phi|\!|\!|_i^2 \\
\geq& \mu_0\|\nabla\phi\|_2^2 - \|\boldsymbol{\beta}\|_\infty\|\nabla\phi\|_2\|\phi\|_2 + \sigma_0\|\phi\|_2^2 \\
\geq& \min\{\mu_0,\sigma_0\}\left(\|\phi\|_2^2 + \|\nabla\phi\|_2^2\right) - \|\boldsymbol{\beta}\|_\infty\left(\frac{\|\phi\|_2^2}{2} + \frac{\|\nabla\phi\|_2^2}{2}\right) \\
\geq& \min\left\{\mu_0 - \frac{\|\boldsymbol{\beta}\|_\infty}{2}, \sigma_0 - \frac{\|\boldsymbol{\beta}\|_\infty}{2}\right\}\|\phi\|_{H^1(\Omega)}^2 \\
\geq& \alpha\|\phi\|_{H^1(\Omega)}^2,
\end{aligned}$$

dove abbiamo usato le diseguaglianze triangolare, di Hölder, di Cauchy-Schwarz e di Young e la definizione di norma in $H^1(\Omega)$ e dove abbiamo inoltre supposto che esistano μ_0, $\{\gamma_{i,0}\}_{i\in\mathcal{I}_R}$ e σ_0 reali tali che $\mu \geq \mu_0 > 0, \sigma \geq \sigma_0$ quasi ovunque in Ω e $\gamma_i \geq \gamma_{i,0} \geq 0$ quasi ovunque su Γ_i per ogni $i \in \mathcal{I}_R$.

La costantedi coercività

$$\alpha = \min\left\{\mu_0 - \frac{\|\boldsymbol{\beta}\|_\infty}{2}, \sigma_0 - \frac{\|\boldsymbol{\beta}\|_\infty}{2}\right\}$$

risulta infine positiva nell'ipotesi che $\mu_0 > \|\boldsymbol{\beta}\|_\infty/2$ e che $\sigma_0 \geq \|\boldsymbol{\beta}\|_\infty/2$.

Dimostrazione della continuità di F Per ogni ψ in $H^1(\Omega)$ si ha:

$$\begin{aligned}
|F\psi| \leq& |\langle f,\psi\rangle| + \eta\sum_{\mathcal{I}_D}|\langle\!\langle g_i,\psi\rangle\!\rangle_i| + \sum_{\mathcal{I}_N}|\langle\!\langle \mu h_i,\psi\rangle\!\rangle_i| + \sum_{\mathcal{I}_R}|\langle\!\langle \mu r_i,\psi\rangle\!\rangle_i| \\
\leq& \|f\|_2\|\psi\|_{H^1(\Omega)} + \eta\sum_{\mathcal{I}_D}|\!|\!|g_i|\!|\!|_i\|\psi\|_{H^1(\Omega)} + \|\mu\|_\infty\sum_{\mathcal{I}_N}|\!|\!|h_i|\!|\!|_i\|\psi\|_{H^1(\Omega)} \\
&+ \|\mu\|_\infty\sum_{\mathcal{I}_R}|\!|\!|r_i|\!|\!|_i\|\psi\|_{H^1(\Omega)} \\
\leq& \left[\|f\|_2 + \eta M_D + \|\mu\|_\infty M_N + \|\mu\|_\infty M_R\right]\|\psi\|_{H^1(\Omega)} \\
\leq& \left[\|f\|_2 + \eta M_D + \|\mu\|_\infty M_N + \|\mu\|_\infty M_R\right]\|\psi\|_{H^1(\Omega)},
\end{aligned}$$

dove abbiamo usato le diseguaglianze triangolare, di Hölder, di Cauchy-Schwarz e di Young e la definizione di norma in $H^1(\Omega)$.

La costante di continuità vale pertanto

$$M = \|f\|_2 + \eta M_D + \|\mu\|_\infty M_N + \|\mu\|_\infty M_R,$$

avendo posto $M_D = \sum_{\mathcal{I}_D}|\!|\!|g_i|\!|\!|_i$, $M_N = \sum_{\mathcal{I}_N}|\!|\!|h_i|\!|\!|_i$ e $M_R = \sum_{\mathcal{I}_R}|\!|\!|r_i|\!|\!|_i$.

1.3 Metodo agli elementi spettrali

Introduciamo ora il metodo agli elementi spettrali (SEM, Spectral Elements Method) per la risoluzione del problema debole (1.5). Esso è un metodo di Galerkin e consiste, pertanto, nell'approssimare lo spazio $H^1(\Omega)$ dove cercare la

soluzione del problema con una opportuna famiglia di suoi sottospazi finitodimensionali $\{V_N\}_N$. la quale garantisca che la soluzione $u_N \in V_N$ del problema approssimato:

trovare $u_N \in V_N$ tale che $\forall v_N \in V_N$

$$a(u_N, v_n) = F v_N \tag{1.6}$$

converga per $N \to \infty$ alla soluzione esatta $u \in H^1(\Omega)$ del problema debole (1.5).

1.3.1 Spazio delle funzioni agli elementi spettrali

Introduciamo ora la famiglia di sottospazi finitodimensionali impiegata nel SEM. Indichiamo con $\{\Omega_k\}_{k=1}^M$ una quadrangolazione in M elementi del dominio Ω, ovvero un insieme di quadrangoli $\Omega_k \subset \mathbb{R}^2$ tali che la loro unione sia l'intero dominio:

$$\bigcup_{k=1}^{M} \Omega_k = \Omega$$

e che la loro intersezione sia vuota, un lato oppure un vertice

$$\Omega_k \cap \Omega_{k'} = \begin{cases} \varnothing \\ P_{k,i} \\ \Gamma_{k,i} \end{cases} \qquad 0 \le k < k' \le M,$$

dove con $P_{k,i}$ e $\Gamma_{k,i}$ $(i = 1, 2, 3, 4)$ indichiamo i vertici ed i lati rispettivamente del quadrangolo Ω_k.

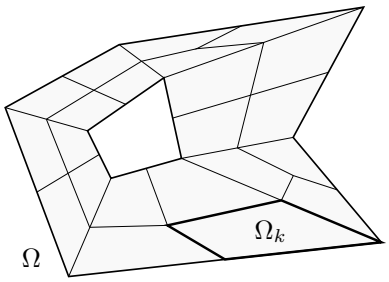

Figura 1.2: Esempio di scomposizione del dominio in quadrangoli

Poniamo $\widehat{\Omega} = [-1, 1] \times [-1, 1] \in \mathbb{R}^2$ essere il rettangolo di riferimento canonico e ne indichiamo con $\widehat{P}_i$ i vertici:

$$\widehat{P}_1 = (-1, -1)^T, \qquad \widehat{P}_2 = (1, -1)^T,$$

$$\widehat{P}_3 = (1, 1)^T, \qquad \widehat{P}_4 = (-1, 1)^T;$$

e con $\widehat{\Gamma}_i$ i lati in senso antiorario:

$$\widehat{\Gamma}_1 = [-1, 1] \times \{-1\}, \qquad \widehat{\Gamma}_2 = \{1\} \times [-1, 1],$$

$$\widehat{\Gamma}_3 = [-1,1]\times\{1\}, \qquad \widehat{\Gamma}_4 = \{-1\}\times[-1,1].$$

Fissiamo per ogni elemento Ω_k una trasformazione di $\widehat{\Omega}$ in Ω_k ovvero sia, per ogni $1 \le k \le M$, $\boldsymbol{\phi}_k : \widehat{\Omega} \to \Omega_k$ un omeomorfismo tale per cui:

$$\boldsymbol{\phi}_k(\widehat{\Omega}) = \Omega_k, \tag{1.7a}$$

$$\boldsymbol{\phi}_k(\widehat{\Gamma}_i) = \Gamma_{k,i} \qquad (i = 1,2,3,4), \tag{1.7b}$$

$$\boldsymbol{\phi}_k(\widehat{P}_i) = P_{k,i} \qquad (i = 1,2,3,4). \tag{1.7c}$$

Definiamo inoltre lo spazio $\mathbb{Q}_N$ delle funzioni polinomiali su $\mathbb{R}^2$ a coefficienti reali di grado minore o uguale ad N rispetto a ciascuna variabile

$$\mathbb{Q}_N = \left\{ v_N : \mathbb{R}^2 \to \mathbb{R} : \exists (a_{k,m}) \in \mathbb{R}^{N\times N}, v_N(\mathbf{x}) = \sum_{i,j=0}^{N} a_{k,m} x^i y^j, \forall \mathbf{x} \in \mathbb{R}^2 \right\}.$$

Nel caso del SEM nella formulazione di Galerkin (1.6) la seguente famiglia di sottospazi finitodimensionali di $H^1(\Omega)$

$$V_N := \left\{ v_N \in C^0(\overline{\Omega}) : v_N|_{\Omega_k} \circ \boldsymbol{\phi}_k \in \mathbb{Q}_N \right\}$$

che chiameremo *spazio delle funzioni agli elementi spettrali su grado N sulla partizione $\{\Omega_k\}$ di Ω.*

1.3.2 Trasformazioni

Risulta pertanto fondamentale nell'implementazione del metodo MES la scelta delle trasformazioni utilizzate per mappare ciascun elemento Ω_k.

In questo progetto scegliamo di addottare trasformazioni bilineari ovvero che possano essere scritte nella forma $\boldsymbol{\phi}(x,y) = \mathbf{A} + \mathbf{B}x + \mathbf{C}y + \mathbf{D}xy$ con $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ e $\mathbf{D}$ costanti in $\mathbb{R}^2$. Se imponiamo le condizioni (1.7c) possiamo definire per $k = 1, \dots, M$

$$\begin{array}{rccl} \boldsymbol{\phi}_k : & \widehat{\Omega} & \to & \Omega_k \\ & (\hat{x},\hat{y})^T & \mapsto & \mathbf{P}_k - \mathbf{Q}_k\hat{x} - \mathbf{R}_k\hat{y} + \mathbf{S}_k\hat{x}\hat{y} \end{array} \tag{1.8}$$

dove:

$$\mathbf{P}_k = \frac{P_{k,1}+P_{k,2}+P_{k,3}+P_{k,4}}{4}, \qquad \mathbf{Q}_k = \frac{P_{k,1}-P_{k,2}-P_{k,3}+P_{k,4}}{4},$$
$$\mathbf{R}_k = \frac{P_{k,1}+P_{k,2}-P_{k,3}-P_{k,4}}{4}, \qquad \mathbf{S}_k = \frac{P_{k,1}-P_{k,2}+P_{k,3}-P_{k,4}}{4}.$$

1.3.2.1 Jacobiano

È immediato calcolare la matrice Jacobiana $J_{\boldsymbol{\phi}_k}$ di $\boldsymbol{\phi}_k$

$$J_{\boldsymbol{\phi}_k}(\hat{x},\hat{y}) = (-\mathbf{Q}_k + \mathbf{S}_k\hat{y}, -\mathbf{R}_k + \mathbf{S}_k\hat{x})$$

così come il suo jacobiano

$$\begin{aligned} |J_{\boldsymbol{\phi}_k}(\hat{x},\hat{y})| &= x_{\mathbf{Q}_k} y_{\mathbf{R}_k} - x_{\mathbf{R}_k} y_{\mathbf{Q}_k} + \hat{x}(x_{\mathbf{S}_k} y_{\mathbf{Q}_k} - x_{\mathbf{Q}_k} y_{\mathbf{S}_k}) + \hat{y}(x_{\mathbf{R}_k} y_{\mathbf{S}_k} - x_{\mathbf{S}_k} y_{\mathbf{R}_k}) \\ &= \alpha_{\mathbf{Q}_k\mathbf{R}_k} - \hat{x}\alpha_{\mathbf{Q}_k\mathbf{S}_k} + \hat{y}\alpha_{\mathbf{R}_k\mathbf{S}_k}, \end{aligned} \tag{1.9}$$

dove $\alpha_{\mathbf{PQ}} := |\mathbf{P},\mathbf{Q}|$.

1.3.2.2 Trasformazione inversa

Più complesso è calcolare la trasformazione $\hat{\boldsymbol{\phi}}_k$ di $\boldsymbol{\phi}_k$.

Caso generale Invertendo (1.8) si ottiene la seguente espressione per la trasformazione inversa

$$\begin{array}{rccc} \hat{\boldsymbol{\phi}}_k: & \Omega_k & \to & \hat{\Omega} \\ & \begin{pmatrix} x \\ y \end{pmatrix} & \mapsto & \begin{pmatrix} \frac{y_{\mathbf{S}_k} x - x_{\mathbf{S}_k} y - a_k + \sqrt{\hat{\psi}_k(x,y)}}{c_k} \\ \frac{y_{\mathbf{S}_k} x - x_{\mathbf{S}_k} y - b_k - \sqrt{\hat{\psi}_k(x,y)}}{d_k} \end{pmatrix} \end{array}$$

dove:

$$\begin{aligned} a_k &= y_{\mathbf{S}_k} x_{\mathbf{P}_k} - x_{\mathbf{S}_k} y_{\mathbf{P}_k} + y_{\mathbf{R}_k} x_{\mathbf{Q}_k} - y_{\mathbf{Q}_k} x_{\mathbf{R}_k} \\ &= \alpha_{\mathbf{P}_k \mathbf{S}_k} + \alpha_{\mathbf{Q}_k \mathbf{R}_k}, \end{aligned}$$

$$\begin{aligned} b_k &= y_{\mathbf{S}_k} x_{\mathbf{P}_k} - x_{\mathbf{S}_k} y_{\mathbf{P}_k} - y_{\mathbf{R}_k} x_{\mathbf{Q}_k} + y_{\mathbf{Q}_k} x_{\mathbf{R}_k} \\ &= \alpha_{\mathbf{P}_k \mathbf{S}_k} - \alpha_{\mathbf{Q}_k \mathbf{R}_k}, \end{aligned}$$

$$\begin{aligned} c_k &= 2(y_{\mathbf{S}_k} x_{\mathbf{Q}_k} - y_{\mathbf{Q}_k} x_{\mathbf{S}_k}) \\ &= 2\alpha_{\mathbf{Q}_k \mathbf{S}_k}, \end{aligned} \qquad \begin{aligned} d_k &= 2(y_{\mathbf{S}_k} x_{\mathbf{R}_k} - y_{\mathbf{R}_k} x_{\mathbf{S}_k}) \\ &= 2\alpha_{\mathbf{R}_k \mathbf{S}_k} \end{aligned}$$

e

$$\hat{\psi}(x,y) = (y_S x - x_S y)^2 + \hat{\psi}_{kx} x - \hat{\psi}_{ky} y + \hat{\psi}_{k0},$$

avendo posto:

$$\begin{aligned} \hat{\psi}_{kx} =& 2\Big[y_{\mathbf{S}_k}(y_{\mathbf{P}_k} x_{\mathbf{S}_k} - y_{\mathbf{S}_k} x_{\mathbf{P}_k}) + y_{\mathbf{R}_k}(y_{\mathbf{S}_k} x_{\mathbf{Q}_k} - x_{\mathbf{S}_k} y_{\mathbf{Q}_k}) \\ &+ y_{\mathbf{Q}_k}(x_{\mathbf{R}_k} y_{\mathbf{S}_k} - x_{\mathbf{S}_k} y_{\mathbf{R}_k}) \Big] \\ =& 2(-y_{\mathbf{S}_k} \alpha_{\mathbf{P}_k \mathbf{S}_k} + y_{\mathbf{R}_k} \alpha_{\mathbf{Q}_k \mathbf{S}_k} + y_{\mathbf{Q}_k} \alpha_{\mathbf{R}_k \mathbf{S}_k}). \end{aligned}$$

$$\begin{aligned} \hat{\psi}_{ky} =& 2\Big[x_{\mathbf{S}_k}(y_{\mathbf{P}_k} x_{\mathbf{S}_k} - y_{\mathbf{S}_k} x_{\mathbf{P}_k}) + x_{\mathbf{R}_k}(y_{\mathbf{S}_k} x_{\mathbf{Q}_k} - x_{\mathbf{S}_k} y_{\mathbf{Q}_k}) \\ &+ x_{\mathbf{Q}_k}(x_{\mathbf{R}_k} y_{\mathbf{S}_k} - x_{\mathbf{S}_k} y_{\mathbf{R}_k}) \Big] \\ =& 2(-x_{\mathbf{S}_k} \alpha_{\mathbf{P}_k \mathbf{S}_k} + x_{\mathbf{R}_k} \alpha_{\mathbf{Q}_k \mathbf{S}_k} + x_{\mathbf{Q}_k} \alpha_{\mathbf{R}_k \mathbf{S}_k}), \\ \hat{\psi}_{k0} =& (y_{\mathbf{S}_k} x_{\mathbf{P}_k} - y_{\mathbf{P}_k} x_{\mathbf{S}_k} + y_{\mathbf{Q}_k} x_{\mathbf{R}_k} - y_{\mathbf{R}_k} x_{\mathbf{Q}_k})^2 \\ &+ 4(y_{\mathbf{Q}_k} x_{\mathbf{P}_k} - y_{\mathbf{P}_k} x_{\mathbf{Q}_k})(y_{\mathbf{R}_k} x_{\mathbf{S}_k} - y_{\mathbf{S}_k} x_{\mathbf{R}_k}) \\ =& (\alpha_{\mathbf{P}_k \mathbf{S}_k} - \alpha_{\mathbf{Q}_k \mathbf{R}_k})^2 - 4(\alpha_{\mathbf{P}_k \mathbf{Q}_k} \alpha_{\mathbf{R}_k \mathbf{S}_k}). \end{aligned}$$

Introducendo le quantità $\alpha_{\mathbf{x}\mathbf{Q}_k}$, $\alpha_{\mathbf{x}\mathbf{R}_k}$ e $\alpha_{\mathbf{x}\mathbf{S}_k}$, dipendenti da $\mathbf{x}$, possiamo riscrivere $\hat{\boldsymbol{\phi}}_k$ in funzione di queste variabili

$$\hat{\boldsymbol{\phi}}_k(\mathbf{x}) = \begin{pmatrix} \frac{\alpha_{\mathbf{x}\mathbf{S}_k} - \alpha_{\mathbf{P}_k \mathbf{S}_k} - \alpha_{\mathbf{Q}_k \mathbf{R}_k} + \sqrt{\hat{\psi}_k(\mathbf{x})}}{2\alpha_{\mathbf{Q}_k \mathbf{S}_k}} \\ \frac{\alpha_{\mathbf{x}\mathbf{S}_k} - \alpha_{\mathbf{P}_k \mathbf{S}_k} + \alpha_{\mathbf{Q}_k \mathbf{R}_k} - \sqrt{\hat{\psi}_k(\mathbf{x})}}{2\alpha_{\mathbf{R}_k \mathbf{S}_k}} \end{pmatrix},$$

essendo

$$\begin{aligned}\hat{\psi}_k(\mathbf{x}) =&(\alpha_{\mathbf{x}\mathbf{S}_k} - \alpha_{\mathbf{P}_k\mathbf{S}_k} + \alpha_{\mathbf{Q}_k\mathbf{R}_k})^2 \\ &+ 2(\alpha_{\mathbf{x}\mathbf{R}_k}\alpha_{\mathbf{Q}_k\mathbf{S}_k} + \alpha_{\mathbf{x}\mathbf{Q}_k}\alpha_{\mathbf{R}_k\mathbf{S}_k} - \alpha_{\mathbf{x}\mathbf{S}_k}\alpha_{\mathbf{Q}_k\mathbf{R}_k}) \\ &- 4\alpha_{\mathbf{P}_k\mathbf{Q}_k}\alpha_{\mathbf{R}_k\mathbf{S}_k}.\end{aligned}$$

Dobbiamo osservare che questa trasformazione inversa è definita solo per gli elementi Ω_k tali per cui le quantità $\alpha_{\mathbf{Q}_k\mathbf{S}_k}$ e $\alpha_{\mathbf{R}_k\mathbf{S}_k}$ sono non nulle, ovvero se l'elemento Ω_k non ha lati paralleli.

Caso trapezio Ora osserviamo che $\alpha_{\mathbf{Q}_k\mathbf{S}_k}$ è il determinante della matrice ottenuta giustapponendo i vettori $P_{k,2} - P_{k,1}$ e $P_{k,4} - P_{k,3}$ ed è quindi nullo solo se questi sono paralleli, ipotizziamo infatti per ipotesi che valga $P_{k,i} \neq P_{k,j}$, $0 \leq i < j \leq 4$.

In questo caso la trasformazione inversa si scrive:

$$\begin{cases} \hat{x} = \frac{x_{\mathbf{Q}_k}(y_{\mathbf{R}_k}x - x_{\mathbf{R}_k}y + x_{\mathbf{R}_k}y_{\mathbf{P}_k} - x_{\mathbf{P}_k}y_{\mathbf{R}_k})}{x_{\mathbf{S}_k}[y_{\mathbf{Q}_k}x - x_{\mathbf{Q}_k}y + x_{\mathbf{Q}_k}y_{\mathbf{P}_k} - x_{\mathbf{P}_k}y_{\mathbf{Q}_k}] + x_{\mathbf{Q}_k}[x_{\mathbf{R}_k}y_{\mathbf{Q}_k} - x_{\mathbf{Q}_k}y_{\mathbf{R}_k}]} \\ \\ \hat{y} = \frac{yx_{\mathbf{Q}_k} - xy_{\mathbf{Q}_k} + x_{\mathbf{P}_k}y_{\mathbf{Q}_k} - y_{\mathbf{P}_k}x_{\mathbf{Q}_k}}{x_{\mathbf{R}_k}y_{\mathbf{Q}_k} - x_{\mathbf{Q}_k}y_{\mathbf{R}_k}} \end{cases}$$

ovvero:

$$\begin{cases} \hat{x} = \frac{x_{\mathbf{Q}_k}(\alpha_{\mathbf{x}\mathbf{R}_k} - \alpha_{\mathbf{P}_k\mathbf{R}_k})}{x_{\mathbf{S}_k}(\alpha_{\mathbf{x}\mathbf{Q}_k} - \alpha_{\mathbf{P}_k\mathbf{Q}_k}) - x_{\mathbf{Q}_k}\alpha_{\mathbf{Q}_k\mathbf{R}_k}} \\ \\ \hat{y} = \frac{\alpha_{\mathbf{x}\mathbf{Q}_k} - \alpha_{\mathbf{P}_k\mathbf{Q}_k}}{\alpha_{\mathbf{Q}_k\mathbf{R}_k}} \end{cases}.$$

Analogamente osserviamo che $\alpha_{\mathbf{R}_k\mathbf{S}_k}$ è il determinante della matrice ottenuta giustapponendo i vettori $P_{k,3} - P_{k,2}$ e $P_{k,1} - P_{k,4}$ ed è quindi nullo solo se questi sono paralleli. In questo caso la trasformazione inversa si scrive:

$$\begin{cases} \hat{x} = \frac{\alpha_{\mathbf{P}_k\mathbf{R}_k} - \alpha_{\mathbf{x}\mathbf{R}_k}}{\alpha_{\mathbf{Q}_k\mathbf{R}_k}} \\ \\ \hat{y} = \frac{x_{\mathbf{R}_k}(\alpha_{\mathbf{P}_k\mathbf{Q}_k} - \alpha_{\mathbf{x}\mathbf{Q}_k})}{x_{\mathbf{S}_k}(\alpha_{\mathbf{P}_k\mathbf{R}_k} - \alpha_{\mathbf{x}\mathbf{R}_k}) - x_{\mathbf{R}_k}\alpha_{\mathbf{Q}_k\mathbf{R}_k}} \end{cases}.$$

Caso parallelogramma Se invece Ω_k è un parallelogramma l'inversa si scrive:

$$\begin{cases} \hat{x} = \frac{\alpha_{\mathbf{P}_k\mathbf{R}_k} - \alpha_{\mathbf{x}\mathbf{R}_k}}{\alpha_{\mathbf{Q}_k\mathbf{R}_k}} \\ \\ \hat{y} = \frac{\alpha_{\mathbf{x}\mathbf{Q}_k} - \alpha_{\mathbf{P}_k\mathbf{Q}_k}}{\alpha_{\mathbf{Q}_k\mathbf{R}_k}} \end{cases}.$$

1.3.2.3 Jacobiano inverso

Per il teorema della funzione inversa la matrice jacobiana può essere ottenuta con il metodo dei complementi algebrici da (1.9)

$$J^{-T}_{\boldsymbol{\phi}_k}(\hat{x}, \hat{y}) = \begin{pmatrix} \frac{\hat{x}y_S - y_R}{\alpha_{QR} - \hat{x}\alpha_{QS} + \hat{y}\alpha_{RS}} & \frac{x_R - \hat{x}x_S}{\alpha_{QR} - \hat{x}\alpha_{QS} + \hat{y}\alpha_{RS}} \\ \frac{y_Q - \hat{y}y_S}{\alpha_{QR} - \hat{x}\alpha_{QS} + \hat{y}\alpha_{RS}} & \frac{\hat{y}x_S - x_Q}{\alpha_{QR} - \hat{x}\alpha_{QS} + \hat{y}\alpha_{RS}} \end{pmatrix}. \tag{1.11}$$

1.3.3 Integrazione numerica

Indichiamo con $\hat{x}_i$ $(i = 0, \ldots, N)$ i nodi di Gauss-Legendre-Lobatto (GLL) sull'intervallo $[-1, 1]$ ovvero l'insieme delle $N-1$ radici della derivata, D_N, del polinomio di Legendre di grado N definito ricorsivamente da:

$$\begin{cases} L_0 = 1 \\ L_1 = x \\ L_{k+1} = \frac{2k+1}{k+1} x L_k - \frac{k}{k+1} L_{k-1} \quad , \quad k > 1 \end{cases}$$

e dagli estremi -1 e 1 ordinati in modo che

$$-1 = \hat{x}_0 < \hat{x}_1 < \cdots < \hat{x}_{N-1} < \hat{x}_N = 1.$$

Definiamo poi i pesi α_i

$$\alpha_i = \frac{2}{N(N+1)} \frac{1}{L_N^2(\hat{x}_i)}.$$

Introduciamo ora la formula di quadratura GLL per il prodotto scalare di $L^2(\Omega)$

$$\langle \phi, \psi \rangle_N := \sum_k^M {}_1 \sum_{ij}^N {}_0 \alpha_{ij}^{(k)} \phi(\mathbf{x}_{ij}^{(k)}) \psi(\mathbf{x}_{ij}^{(k)}) \qquad \forall \phi, \psi \in C^0(\overline{\Omega}, \mathbb{R}),$$

$$\langle \boldsymbol{\phi}, \boldsymbol{\psi} \rangle_N := \sum_k^M {}_1 \sum_{ij}^N {}_0 \alpha_{ij}^{(k)} \boldsymbol{\phi}(\mathbf{x}_{ij}^{(k)}) \cdot \boldsymbol{\psi}(\mathbf{x}_{ij}^{(k)}) \qquad \forall \boldsymbol{\phi}, \boldsymbol{\psi} \in C^0(\overline{\Omega}, \mathbb{R}^2),$$

dove i nodi $\mathbf{x}_{ij}^{(k)}$ ed i relativi pesi $\alpha_{ij}^{(k)}$ sono così definiti:

$$\mathbf{x}_{ij}^{(k)} := \boldsymbol{\phi}_k(\hat{x}_i, \hat{x}_j), \qquad \alpha_{ij}^{(k)} := |J_{\boldsymbol{\phi}_k}(\hat{x}_i, \hat{x}_j)|.$$

La formula di quadratura GLL per il prodotto scalare di $L^2(\Gamma_i)$ è invece

$$\langle\!\langle \phi, \psi \rangle\!\rangle_{i,N} := \Big(\sum_{\mathbf{x}_{ij}^k \in \Gamma_i} \alpha_{ij}^{(k)} \Big)^{-1} \sum_{\mathbf{x}_{ij}^k \in \Gamma_i} \alpha_{ij}^{(k)} \phi(\mathbf{x}_{ij}^{(k)}) \psi(\mathbf{x}_{ij}^{(k)}) \qquad \forall \phi, \psi \in C^0(\Gamma_i)$$

esendo k l'unico indice per cui $\Gamma_i \subset \Omega_k$.

Definiamo le seguenti approssimazioni della forma a

$$\begin{aligned} a_N(\phi, \psi) :=& \langle \mu \nabla \phi, \nabla \psi \rangle_N + \langle \boldsymbol{\beta} \cdot \nabla \phi, \psi \rangle_N + \langle \sigma \phi, \psi \rangle_N + \\ &+ \eta \sum_{i \in \mathcal{I}_D} \langle\!\langle \phi, \psi \rangle\!\rangle_{i,N} + \sum_{i \in \mathcal{I}_R} \langle\!\langle \mu \gamma_i u, v \rangle\!\rangle_{i,N} \end{aligned}$$

e del funzionale F

$$F_N \psi := \langle f, \psi \rangle_N + \eta \sum_{\mathcal{I}_D} \langle\!\langle g_i, \psi \rangle\!\rangle_{i,N} + \sum_{\mathcal{I}_N} \langle\!\langle \mu h_i, \psi \rangle\!\rangle_{i,N} + \sum_{\mathcal{I}_R} \langle\!\langle \mu r_i, \psi \rangle\!\rangle_{i,N}.$$

e consideriamo il seguente problema di Galerkin generalizzato:

trovare $\widetilde{u}_N \in V_N$ tale che $\forall v_N \in V_N$

$$a_N(\widetilde{u}_N, v_n) = F_N v_N \tag{1.12}$$

1.3.4 Formulazione algebrica

Sia $\{\psi_I\}_{I=1}^N$ la base lagrangiana per lo spazio V_N degli elementi spettrali di grado d su una quadrangolazione di Ω in M sottodomini allora il problema approssimato agli elementi spettrali con integrazione numerica (1.12) è equivalente al sistema $\forall I = 0, \ldots, N$

$$\begin{aligned}\langle \mu\nabla u, \nabla\psi_I\rangle_N + \langle \boldsymbol{\beta}\cdot\nabla u, \psi_I\rangle_N + \langle \sigma u, \psi_I\rangle_N + \sum_{i\in\mathcal{I}_D} \eta\langle\!\langle u, \psi_I\rangle\!\rangle_{i,N} + \sum_{i\in\mathcal{I}_R} \langle\!\langle \mu\gamma_i u, \psi_I\rangle\!\rangle_{i,N} \\ = \langle f\psi_I\rangle_N + \sum_{i\in\mathcal{I}_D} \eta\langle\!\langle g, \psi_I\rangle\!\rangle_{i,N} + \sum_{i\in\mathcal{I}_N} \langle\!\langle \mu h_i, \psi_I\rangle\!\rangle_{i,N} + \sum_{i\in\mathcal{I}_R} \langle\!\langle \mu r_i, \psi_I\rangle\!\rangle_{i,N}.\end{aligned}$$

Poiché $u \in V_N$ possiamo scrivere $u = \sum_{I'=1}^N u_{I'}\psi_{I'}$, otteniamo così il sistema

$$\begin{aligned}\sum_{I'}{}_1^N u_{I'}\langle \mu\nabla\psi_{I'}, \nabla\psi_I\rangle_N + \sum_{I'}{}_1^N u_{I'}\langle \boldsymbol{\beta}\cdot\nabla\psi_{I'}, \psi_I\rangle_N + \sum_{I'}{}_1^N u_{I'}\langle \sigma\psi_{I'}, \psi_I\rangle_N \\ + \sum_{I'}{}_1^N u_{I'} \sum_{i\in\mathcal{I}_D} \eta\langle\!\langle \psi_{I'}, \psi_I\rangle\!\rangle_{i,N} + \sum_{I'}{}_1^N u_{I'} \sum_{i\in\mathcal{I}_R} \langle\!\langle \mu\gamma_i\psi_{I'}, \psi_I\rangle\!\rangle_{i,N} \\ = \langle f\psi_I\rangle_N + \sum_{i\in\mathcal{I}_D} \eta\langle\!\langle g_i, \psi_I\rangle\!\rangle_{i,N} + \sum_{i\in\mathcal{I}_N} \langle\!\langle \mu h_i, \psi_I\rangle\!\rangle_{i,N} + \sum_{i\in\mathcal{I}_R} \langle\!\langle \mu r_i, \psi_I\rangle\!\rangle_{i,N},\end{aligned}$$

che scritto in forma matriciale diviene

$$[A^d + A^c + A^r + A^b]\mathbf{u} = [\mathbf{f} + \mathbf{f}^b],$$

dove le matrici $A^d = (a^d_{II'})$, $A^c = (a^c_{II'})$, $A^r = (a^r_{II'})$ e $A^b = (a^b_{II'})$ sono definite come segue:

$$\begin{aligned} a^d_{II'} &= \langle \mu\nabla\psi_{I'}, \nabla\psi_I\rangle_N, \quad & a^c_{II'} &= \langle \boldsymbol{\beta}\cdot\nabla\psi_{I'}, \psi_I\rangle_N, \\ a^r_{II'} &= \langle \sigma\psi_{I'}, \psi_I\rangle_N, & a^b_{II'} &= \sum_{i\in\mathcal{I}_D} \eta\langle\!\langle \psi_{I'}, \psi_I\rangle\!\rangle_{i,N} + \sum_{i\in\mathcal{I}_R} \langle\!\langle \mu\gamma_i\psi_{I'}, \psi_I\rangle\!\rangle_{i,N} \end{aligned}$$

ed i vettori $\mathbf{f} = (f_I)$ ed $\mathbf{f}^b = (f^b_I)$ sono definiti come segue:

$$f_I = \langle f, \psi_I\rangle_N, \qquad f^b_I = \sum_{i\in\mathcal{I}_N} \langle\!\langle \mu h_i, \psi_I\rangle\!\rangle_{i,N} + \sum_{i\in\mathcal{I}_R} \langle\!\langle \mu r_i, \psi_I\rangle\!\rangle_{i,N}.$$

Capitolo 2

Implementazione

Per l'implementazione del SEM-NI abbiamo deciso di utilizzare il linguaggio di programmazione C++ orientato agli oggetti. Abbiamo deciso inoltre di suddividere il progetto in più librerie in modo da enfatizzare la struttura del metodo e le fasi in cui questo si articola:

PreProcessing In questa fase viene impostato il problema, vengono cioè create quelle strutture di dati necessarie alle successive fasi. Questa fase non costituisce propriamente parte dell'implementazione SEM-NI essendo in realtà preparatoria;

Assembling In questa fase vengono elaborati i dati geometrici e funzionali in ingresso e costuite le matrici algebriche associate al problema da risolvere;

Solving Questa fase consiste nel risolvere il sistema algebrico costruito in fase di Assembling;

PostProcessing Questa costituisce l'ultima fase del progetto. In questa fase viene elaborato il risultato algebrico per ottenere risultati sintetici in forma utilizzabile per compiere un'analisi del risultato ottenuto.

Il progetto implementa inoltre una interfaccia grafica (GUI) sia per l'inserimento degli input che per la visualizzazione del risultato ed una libreria con le implementazioni delle funzionalità di input/outup su file.

2.1 Dipendenze

Boost Le librerie C++ Boost[1] sono un insieme di librerie free software che estendono le funzionalità del C++ . La maggior parte delle librerie Boost è distribuita con la Boost Software License, che permette agli sviluppatori di utilizzarle in progetti sia free che proprietari.

CGAL (Computational Geometry Algorithms Library) L'obiettivo del CGAL Open Source Project[2] è permettere un facile accesso ad algoritmi geometrici efficienti ed affidabili in forma di libreria C++ . È distribuita con licenza LGPL/QPL.

[1] Website `http://www.boost.org/`. Una versione precompilata per Windows può essere trovata su `http://www.boostpro.com`

[2] Website `http://www.cgal.org/`

TNT/JAMA (Template Numerical Toolkit[3]) è una libreria software per la manipolazione di vettori e matrici in C++ sviluppato dallo U.S. National Institute of Standards and Technology, NIST. TNT è l'analogo della libreria BLAS utilizzata da LAPACK. Gli algoritmi di livello superiore sono implementati nella libreria JAMA, sviluppata anch'essa al NIST, la quale utilizza TNT. TNT e JAMA sono di pubblico dominio.

Libarchive[4] Libarchive È costituito da una libreria C e da alcuni stumenti da linea di comando per la lettura e la scrittura di archivi, compressi e non, sviluppata da FreeBSD e mantenuta da Google. È distribuito con licenza New BSD License.

Qt[5] una libreria multipiattaforma per lo sviluppo di programmi con interfaccia grafica tramite l'uso di widget. Qt, ampiamente utilizzato nell'ambiente desktop KDE, viene sviluppato dall'azienda Qt Software ed è rilasciato sotto doppia licenza (GPL e commerciale).

QtMmlWidget[6] è un componente per la visualizzazione di formule matematiche scritte in MathML 2.0. È distribuito con licenza LGPL.

Qwt (Qt Widgets for Technical Applications[7]) È un insieme di Qt widget, componenti GUI e utilità sviluppati proncipalmente per l'utilizzo in programmi tecnici. Oltre a un widget per grafici bidimensionali fornisce scale, slider, bussole, termometri, manopole ed altri strumenti per controllare o visualizzare singoli valori, array, o intervalli di dati di tipo `double`. È distribuito con Qwt License Version 1.0.

QwtPlot3D[8] è una libreria C++ basata su Qt ed OpenGL che fornisce un insieme di widget 3D. È rilasciata sotto licenza Zlib.

2.2 Classi di base

Le varie librerie implementate nel progetto utilizzano delle strutture dati comuni, delle classi, che sono state implementate nella libreria `SemSolver`. Questa libreria è composta interamente da classi template ed è quindi costituita esclusivamente da header senza nessun file binary. Nella libreria, all'interno del namespace `SemSolver`, sono implementati i seguenti oggetti algebrici:

- `Matrix` È una classe template derivata da `TNT::Array2D` che rappresenta le matrici come array dinamici. Il tipo degli elementi di una matrice è un parametro template. Sono implementati, oltre ai metodi della classe base, un metodo per il calcolo dgli autovalori reali e le operazioni di addizione e prodotto.

- `Polynomial` I polinomi sono rappresentati come `std::vector` di coefficienti il cui tipo è un parametro template. Per i polinomi sono implementati le operazioni di addizione, di sottrazione, moltiplicazione, di divisione (long division) e di confronto ed i metodi per il calcolo della derivata, delle radici e del quoziente con il metodo di Ruffini.

[3] Website `http://math.nist.gov/tnt/`
[7] WebSite `http://qwt.sourceforge.net/`

- `Vector` È una classe template derivata da `TNT::Array1D` che rappresenta i vettori come array dinamici. Il tipo dei suoi elementi è un parametro template.

i seguenti oggetti geometrici:

- `Point` È una classe template parametrizzata dalla dimensione e dal tipo delle coordinate. `Point<2,X>` è una classe derivata da `CGAL::Point_2 < CGAL::Filtered_kernel< CGAL::Simple_cartesian<X> > >`. Per i punti sono definiti i seguenti contenitori:
  - `PointsSet` Rappresenta un insieme di `Point`. Ogni elemento è unico e non vi sono duplicati.
  - `PointsMap` Rappresenta una mappa fra `Point` ed elementi di tipo arbitrario. I `Point` sono usati come indici e ad ogni indice corrisponde un solo valore.
  - `PointsBimap` Rappresenta una mappa biiettiva fra `Point` ed `int`. Sia `Point` che `int` possone essere usati come indici.
- `Segment` È una classe template parametrizzata dalla dimensione e dal tipo delle coordinate. `Segment<2,X>` è una classe derivata da `CGAL::Segment_2 < CGAL::Filtered_kernel< CGAL::Simple_cartesian<X> > >`. Per i segmenti è definito il seguente contenitore:
  - `SegmentsMap` Rappresenta una mappa fra `int` e `Segment`. Gli `int` sono usati come indici e ad ogni indice corrisponde un solo `Segment`.
- `Polygon` È una classe template parametrizzata dalla dimensione e dal tipo delle coordinate. `Polygon<2,X>` è una classe derivata da `CGAL:: Polygon_2< CGAL::Filtered_kernel< CGAL::Simple_cartesian<X> > >`. Sono rappresentati cone una catena chiusa di lati.
- `PolygonWithHoles` È una classe template parametrizzata dalla dimensione e dal tipo delle coordinate. `Polygon<2,X>` è una classe derivata da `CGAL ::Polygon_2< CGAL::Filtered_kernel< CGAL::Simple_cartesian<X> > >`. Mentre `Polygon` è una curva chiusa, un `PolygonWithHoles` è una curva chiusa, detta contorno esterno con zero o più curve chiuse, dette contorni interni o buchi, contenute in essa. L'intersezione della parte interna del contorno esterno e delle parti esterne dei buchi costituisce la sua parte interna.
- `Polygonation` Descrive la partizione in `Polygonation::Element` di un dominio descrivibile con un `PolygonWithHoles` semplice, ovvero il cui interno sia semplicemente connesso. Ogni `Polygonation::Element` è descritto da un `Polygon` che ne descrive la geometria e da uno `std::vector <int>` che contiene gli indici relativi agli elementi vicini, ovvero con un lato in comuine all'elemento.
- `SemGeometry` Descrive la geometria di un problema agli elementi spettrali. È costituito da un `PSLG` che ne descrive il dominio Ω e da una `Polygonation` che ne descrive la partizione in elementi quadrangolari $\{\Omega_k\}$.

ed i seguenti oggetti funzionali:

- `Function` È una classe template puramente virtuale per descrive una generica funzione matematica. I parametri template sono i tipi del dominio e del codominio. Ogni classe derivata deve definire il metodo `evaluate` che restituisca il valore assunto dalla funzione in un particolare elemento del dominio. Sono implementate le seguenti classi derivate:
  - `Homeomorphism` È una classe template puramente virtuale per descrivere un omeomorfismo, cioè una funzione continua e biiettiva la cui inversa sia continua. Ogni classe derivata deve definire anche il metodo `evaluateInverse` che restituisca quel particolare elemento del dominio dove la funzione assume un dato valore. È implementata la seguente classe derivata
    - `BilinearTransformation` È la classe utilizzata per descrivere le mappe $\boldsymbol{\phi}_k$ dall'elemento di riferimento $\widehat{\Omega}$ ai singoli elementi Ω_k descritte in §1.3.2.
  - `PolynomialFunction` Descrive le funzioni polinomiali a variabili separate, ovvero le funzioni $p : \mathbb{K}^d \in \mathbb{K}$ tali per cui esistono d polinomi p_i tali che valga $p(\mathbf{x}) = \prod_{i=1}^{d} p_i(x_i), \forall \mathbf{x} = (x_i) \in \mathbb{K}^d$. Il tipo usato per rappresentare gli elementi di $\mathbb{K}$ è un parametro template così come la dimensione d.
  - `ScriptFunction` Permette la definizione delle funzioni in script. In questo modo è possibile utilizzare funzioni definite durante l'esecuzione del codice, ad esempio, tramite file ECMA Script[9] esterni.
  - `SemFunction` Descrive gli elementi, v_N dello spazio agli elementi spettrali V_N per cui $\hat{v}_k = v_N|_{\Omega_k} \circ \boldsymbol{\phi}_k$ è una funzione polinomiale a variabili separate. È la classe usata per descrivere le funzioni di base dello spazio agli elementi spettrali V_N. Ogni funzione agli elementi spettrali è rappresentata da una partizione $\{\Omega_k\}$, dalle trasformazioni $\{\boldsymbol{\phi}_k\}$ e dalle funzioni polinomiali a variabile separate $\{\hat{v}_k\}$.
- `HilbertSpace` È una classe template puramente virtuale per descrivere gli spazi di Hilbert (di dimensione finitta), $(H, \langle \cdot, \cdot \rangle)$. È descritta da uno `std::vector` di funzioni (il cui tipo è un parametro template) che ne rappresenti una base. Ogni sua classe derivata deve definire i metodi `scalarProduct` che ne implementi il prodotto scalare, `norm` che ne implementi la norma indotta dal prodotto scalare e `projection` che ad ogni `Function` associ un `HilbertSpace::Element`, ovvero la sua proiezione rappresentata attraverso le sue coordinate rispetto alla base. È definita la seguente classe derivata:
  - `SemSpace` Implementa lo spazio di Hilbert V_N. È rappresentato da un vettore di `SemFunction` che ne rappresenta la base, un vettore di `SemSpace::Node` e varie mappe che permettono di accedere ai nodi con varie indiciazioni. Ogni nodo `x` può essere accesso attraverso la

[9] ECMAScript è un linguaggio di scripting standardizzato dalla Ecma International nelle specificazioni ECMA-262 e ISO/IEC 16262. È un linguaggio largamente utilizzato nei suoi dialetti, fra i quali JavaScript, JScript e ActionScript. Per comodità i metodi dell'oggetto `Math` possono essere utilizzati direttamente (ad esempio si può scrivere `sin` invece di `Math.sin`

tripla di indici (i, j, k), ad indicare che $\mathbf{x}$ è la trasformazione tramite $\boldsymbol{\phi}_k$ del nodo GLL $(\hat{x}_j, \hat{x}_k)$, o attraverso la coppia di indici (i, j), ovvero $\mathbf{x}$ è il j-esimo nodo sul lato Γ_i di Ω.

nonché le seguenti strutture:

- `PSLG` È una struttura per descrivere uno Planar Straight Line Graph, ovvero un elenco dei vertici, lati e cavità che costituiscono un poligono eventualmente cavo.
- `Equation` È una classe puramente virtuale per descrivere una generica equazione. È implementata la seguente classe derivata:
  - `DiffusionConvectionReactionEquation` Descrive una generica equazione di diffusione-trasporto-reazione stazionaria.
- `BoundaryConditions` Rappresenta le condizioni al contorno di un generico problema differenziale su un dominio poligonale. Ad ogni lato del dominio è possibile associare una specifica condizione di Dirichlet, Neumann o Robin.
- `SemParameters` Sono il grado degli elementi spettrali da utilizzare, la tolleranza (un valore in assoluto minore di questo parametro viene trascurato) ed il coefficiente di penalità η.
- `Problem` Descrive un problema differenziale tramite una `SemGeometry` che ne descriva dominio e sottodomini, una `Equation`, delle `BoundaryConditions` e dei `SemParameters`.

2.3 PreProcessor

Nel namespace `SemSolver::PreProcessor` sono implementati gli algoritmi che permettono la costruzione di una `SemGeometry` a partire da un `PSLG`. In un primo tempo, il progetto implementava questa fase sfruttando la libreria Triangle[10] e CQMesh[11] , in questo modo era possibile ottenere una quadrangolazione a partire da un generico dominio poligonale. A tale scopo avevamo dovuto apportare delle modifiche al codice iniziale di CQMesh per inserirlo nel progetto, ma le quadrangolazioni che si ottenevano erano spesso degeneri, infatti spesso gli elementi ottenuti presentavano lati degeneri.

Nella implementazione corrente, la costruzione della quadrangolazione sfrutta invece la libreria CGAL. Il dominio viene descritto in un file `.poly` che consiste essenzialmente in una lista di vertici e lati ed eventualmente anche cavità; tali file costituivano l'input utilizzato dalla libreria Triangle. Un file `.poly` è costituito da tre parti: una che descrive i vertici costituita da una prima linea

```
<# of vertices> <dimension)> <# of attributes> <# of boundary markers>
```

[10] Triangle permette di costruire triangolazioni di Delaunay, Voronoi diagrams e mesh triangolari. `http://www.cs.cmu.edu/~quake/triangle.html`

[11] CQMesh è un programma C++ per la generazione di mesh quadrangolari a partire da una triangolazione di un dominio poligonale.
`http://www.dimap.ufrn.br/~mfsiqueira/Marcelo_Siqueiras_Web_Spot/cqmesh.html`

seguita da una linea per ogni vertice

```
<vertex id> <x> <y> [attributes] [boundary marker]
```

una parte che ne descrive i lati costituita da una prima linea

```
<# of segments> <# of boundary markers>
```

seguita da una linea per ogni lato

```
<segment id> <endpoint> <endpoint> [boundary marker]
```

e l'ultima parte che ne descrive le cavità costituita da una prima linea

```
<# of holes>
```

seguita da una linea per ogni cavità

```
<hole id> <x> <y>
```

Una volta letto il file `.poly` in input e creato il corrispondente `PSLG`, è possibile generare una poligonazione del dominio associato invocando la funzione `compute_polygonation_from_pslg` che prende come argomenti una referenza costante a `PSLG<X>` e una referenza a `Polygonation<2, X>`. L'algoritmo implementato in questa funzione segue il seguente schema

1. calcola la lista di sequenze di vertici descritti dal `PSLG`, una sequenza per il contorno esterno ed una per ogni cavità;
2. costruisce il `PoligonWithHoles` definito dalle sequenze;
3. costuisce lo scheletro del dominio invocando la funzione `CGAL::create_interior_straight_skeleton_2`
4. costuisce la partizione del dominio individuata dallo scheletro. la partizione così ottenuta non è costituita necessariamente da elementi quadrangolari convessi.

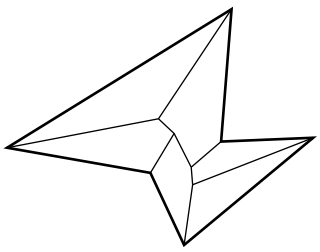

Figura 2.1: Esempio di scheletro

Nel caso in cui la poligonazione non sia una quadrangolazione sarà necessario raffinarla per poterla utilizzare in una `SemGeoemtry`. La raffinazione è ottenuta dividendo ogni elemento nei sottoelementi quadrangolari che si ottengono congiungendo il baricentro con i punti medi dei lati. Questo algoritmo, però, nel caso di elementi non convessi non garantisce che la partizione prodotta sia effettivamente una quadrangolazione valida, in quanto si potrebbero creare sovrapposizioni fra gli elementi.

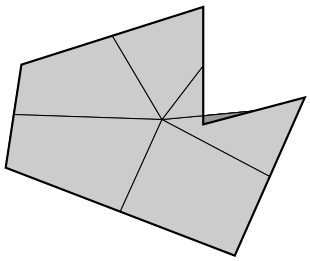

Figura 2.2: Esempio di raffinamento che genera quadrangoli sovrapposti

2.4 Assembler

Per le gestione e manipolazione delle matrici abbiamo utilizzato la libreria TNT, che rappresenta le matrice in forma piena. Questa scelta è ottimale nel caso di pochi sottodomini ma può portare a problemi di memoria nel caso si utilizzino più sottodomini computazionali. Infatti la matrice associata ad un problema di diffusione-trasporto-reazione cui si perviene è piena solo qualora ogni sottodominio sia connesso agli altri.

Osserviamo innanzi tutto che un nodo $\mathbf{x}_I$ può essere indiciato con una tripla di indici (i, j, k) ad indicare che $\mathbf{x}_{jk}^{(i)} = \mathbf{x}_I$, ma che triple differenti possono indicare lo stesso nodo. In paricolare i nodi possono essere classificati così

- nodi interni: $\mathbf{x}_I \in \mathring{\Omega}_i$.
  In tal caso esiste un'unica tripla di indici $(i, j_I^{(i)}, k_I^{(i)})$ tale per cui $\mathbf{x}^{(i)}_{j_I^{(i)} k_I^{(i)}} = \mathbf{x}_I$.

- nodi interni ad un lato di bordo: $\mathbf{x}_I \in \mathring{\Gamma}_{i,\iota} \cap \partial\Omega$.
  In tal caso esiste un'unica tripla di indici $(i, j_I^{(i)}, k_I^{(i)})$ tale per cui $\mathbf{x}^{(i)}_{j_I^{(i)} k_I^{(i)}} = \mathbf{x}_I$.

- nodi interni ad un lato non di bordo: $\mathbf{x}_I \in \mathring{\Gamma}_{i,\iota} \not\subset \partial\Omega$.
  In tal caso esistono due triple di indici $(i_l, j_I^{(i_l)}, k_I^{(i_l)})_{l=1,2}$ tali per cui $\mathbf{x}^{(i_l)}_{j_I^{(i_l)} k_I^{(i_l)}} = \mathbf{x}_I$.

- vertici di bordo: $\mathbf{x}_I = P_{i,\iota} \in \partial\Omega$.
  In tal caso esiste un'unica tripla di indici $(i, j_I^{(i)}, k_I^{(i)})$ tale per cui $\mathbf{x}^{(i)}_{j_I^{(i)} k_I^{(i)}} = \mathbf{x}_I$.

- vertici non di bordo: $\mathbf{x}_I = P_{i,\iota} \notin \partial\Omega$.
  In tal caso esistono più triple di indici $(i_l, j_I^{(i_l)}, {}^{(i_l)}_I)_l$ tali per cui $\mathbf{x}^{(i_l)}_{j_I^{(i_l)} k_I^{(i_l)}} = \mathbf{x}_I$.

Indichiamo ora con $\{\psi_I\}$ la seguente base lagrangiana per V_N nei nodi $\mathbf{x}_I$:

$$\psi_I(\mathbf{x}) = \begin{cases} 1 & \text{se } \mathbf{x} = \mathbf{x}_I \\ \hat{\psi}_{j_I^{(i)} k_I^{(i)}}(\hat{\boldsymbol{\phi}}^{(i)}(\mathbf{x})) & \text{se } \mathbf{x} \in \mathring{\Omega}_i, \mathbf{x}_I \in \Omega_i \\ 0 & \text{altrimenti} \end{cases},$$

dove $\hat{\psi}_{jk}(x,y) := \hat{\psi}_j(x)\hat{\psi}_k(y)$ essendo $\hat{\psi}_j$ il polinomio carratteristico associato al j-esimo nodo GLL, la cui espressione analitica è

$$\psi_j(x) = -\frac{1}{N(N+1)} \frac{(1-x^2)D_N(x)}{(x-x_i)L_N(x_i)}$$

Osserviamo che queste funzioni di base non sono lisce sui lati interni della poligonazione, pertanto ne possiamo definire univocamente il gradiente soltanto delle restrizioni su un particolare elemento Ω_i delle funzione di base introdotte:

$$\begin{aligned} \nabla^{(i)}\psi_I :=& \nabla\psi_I|_{\Omega_i} \\ =& \nabla[\hat{\psi}_{j_l k_l} \circ \hat{\boldsymbol{\phi}}^{(l)}] \\ =& J^T_{\hat{\boldsymbol{\phi}}^{(l)}} \hat{\nabla}\hat{\psi}_{j_l k_l} \\ =& J^{-T}_{\boldsymbol{\phi}^{(l)}} \hat{\nabla}\hat{\psi}_{j_l k_l}. \end{aligned}$$

2.4.1 Costruzione della matrice di diffusione

Indichiamo $\mathcal{L}$ e $\mathcal{L}'$ gli insiemi degli indici relativi ai sottodomini che costituiscono il supporto delle funzioni di base ψ_I e $\psi_{I'}$ rispettivamente e con $\mathcal{L}'' = \mathcal{L} \cap \mathcal{L}'$ la loro intersezione:

$$\mathfrak{supp}(\psi_I) = \bigcup_{l \in \mathcal{L}} \Omega_l, \qquad \mathfrak{supp}(\psi^{I'}) = \bigcup_{l' \in \mathcal{L}'} \Omega_{l'};$$

allora esistono $\{j_I^{(l)}\}_{l\in\mathcal{L}}$, $\{k_I^{(l)}\}_{l\in\mathcal{L}}$, $\{j_I^{(l')}\}_{l'\in\mathcal{L}'}$ e $\{k_I^{(l')}\}_{l'\in\mathcal{L}'}$ tali per cui;

$$\psi_{I|\Omega_l} = \hat{\psi}_{j_I^{(l)} k_I^{(l)}} \circ \hat{\boldsymbol{\phi}}^{(l)}, \forall l \in \mathcal{L}, \qquad \psi_{I'|\Omega_{l'}} = \hat{\psi}_{j_{I'}^{(l')} k_{I'}^{(l')}} \circ \hat{\boldsymbol{\phi}}^{(l')}, \forall l' \in \mathcal{L}';$$

allora la componente sulla I-esima riga della I'-esima colonna della matrice A^d relativa al termine diffusivo si scrive:

$$\begin{aligned} a^d_{II'} &= \langle \mu\nabla\psi_{I'}, \nabla\psi_I \rangle_N \\ &= \sum_i^M \sum_{jk}^N \alpha^{(i)}_{jk} \mu(\mathbf{x}^{(i)}_{jk}) \nabla^{(i)}\psi_{I'}(\mathbf{x}^{(i)}_{jk}) \cdot \nabla^{(i)}\psi_I(\mathbf{x}^{(i)}_{jk}) \\ &= \sum_{l\in\mathcal{L}''} \sum_{jk}^N \alpha^{(l)}_{jk} \mu(\mathbf{x}^{(l)}_{jk}) [J^{-T}_{\boldsymbol{\phi}^{(l)}}(\hat{\mathbf{x}}_{jk}) \hat{\nabla}\hat{\psi}_{j_{I'}^{(l)} k_{I'}^{(l)}}(\hat{\mathbf{x}}_{jk})] \cdot [J^{-T}_{\boldsymbol{\phi}^{(l)}}(\hat{\mathbf{x}}_{jk}) \hat{\nabla}\hat{\psi}_{j_I^{(l)} k_I^{(l)}}(\hat{\mathbf{x}}_{jk})] \\ &= \sum_{l\in\mathcal{L}''} \sum_{jk}^N a^d(l, j_I^{(l)}, k_I^{(l)}, l, j_{I'}^{(l)}, k_{I'}^{(l)}, l, j, k), \end{aligned}$$

dove $a^d(i,j,k,i',j',k',i'',j'',k'') = \alpha^{(i'')}_{j''k''} \mu(\mathbf{x}^{(i'')}_{j''k''}) [J^{-T}_{\boldsymbol{\phi}^{(i')}}(\hat{\mathbf{x}}_{j''k''}) \hat{\nabla}\hat{\psi}_{j'k'}(\hat{\mathbf{x}}_{j''k''})] \cdot [J^{-T}_{\boldsymbol{\phi}^{(i)}}(\hat{\mathbf{x}}_{j''k''}) \hat{\nabla}\hat{\psi}_{jk}(\hat{\mathbf{x}}_{j''k''})]$.

Risulta pertanto che $a^d_{II'} = 0$ se e solo se $\mathfrak{supp}(\psi_I) \cap \mathfrak{supp}(\psi'_I) \neq \varnothing$.

2.4.2 Costruzione della matrice di trasporto

La componente sulla I-esima riga della I'-esima colonna della matrice A^c relativa al termine convettivo si scrive:

$$\begin{aligned}
a^c_{II'} &= \langle \boldsymbol{\beta}\cdot\nabla\psi_{I'},\psi_I\rangle_N \\
&= \sum_i^M{}_1 \sum_{jk}^N{}_0 \alpha^{(i)}_{jk}\boldsymbol{\beta}(\mathbf{x}^{(i)}_{jk})\cdot\nabla^{(i)}\psi_{I'}(\mathbf{x}^{(i)}_{jk})\psi_I(\mathbf{x}^{(i)}_{jk}) \\
&= \sum_{l\in\mathcal{L}''}\sum_{jk}^N{}_0 \alpha^{(l)}_{jk}\boldsymbol{\beta}(\mathbf{x}^{(l)}_{jk})\cdot[J^{-T}_{\boldsymbol{\phi}^{(l)}}(\hat{\mathbf{x}}_{jk})\hat{\nabla}\hat{\psi}_{j^{(l)}_{I'}k^{(l)}_{I'}}(\hat{\mathbf{x}}_{jk})]\hat{\psi}_{j^{(l)}_I k^{(l)}_I}(\hat{\mathbf{x}}_{jk}) \\
&= \sum_{l\in\mathcal{L}''}\alpha^{(l)}_{j^{(l)}_I k^{(l)}_I}\boldsymbol{\beta}(\mathbf{x}^{(l)}_{j^{(l)}_I k^{(l)}_I})\cdot[J^{-T}_{\boldsymbol{\phi}^{(l)}}(\hat{\mathbf{x}}_{j^{(l)}_I k^{(l)}_I})\hat{\nabla}\hat{\psi}_{j^{(l)}_{I'}k^{(l)}_{I'}}(\hat{\mathbf{x}}_{j^{(l)}_I k^{(l)}_I})] \\
&= \sum_{l\in\mathcal{L}''} a^c(l,j^{(l)}_I,k^{(l)}_I,l,j^{(l)}_{I'},k^{(l)}_{I'}),
\end{aligned}$$

dove $a^c(i,j,k,i',j',k') = \alpha^{(i)}_{jk}\boldsymbol{\beta}(\mathbf{x}^{(i)}_{jk})\cdot[J^{-T}_{\boldsymbol{\phi}^{(i')}}(\hat{\mathbf{x}}_{jk})\hat{\nabla}\hat{\psi}_{j'k'}(\hat{\mathbf{x}}_{jk})]$.

Risulta pertanto che $a^c_{II'}=0$ se e solo se $\mathtt{supp}(\psi_I)\cap\mathtt{supp}(\psi'_I)\neq\varnothing$.

2.4.3 Costruzione della matrice di reazione

La componente sulla I-esima riga della I'-esima colonna della matrice A^r relativa al termine reattivo si scrive:

$$\begin{aligned}
a^r_{II'} &= (\gamma\psi_{I'},\psi_I)_N \\
&= \sum_i^M{}_1 \sum_{jk}^N{}_0 \alpha^{(i)}_{jk}\gamma(\mathbf{x}^{(i)}_{jk})\psi_{I'}(\mathbf{x}^{(i)}_{jk})\psi_I(\mathbf{x}^{(i)}_{jk}) \\
&= \sum_{l\in\mathcal{L}''}\sum_{jk}^N{}_0 \alpha^{(l)}_{jk}\gamma(\mathbf{x}^{(l)}_{jk})\hat{\psi}_{j^{(l)}_{I'}k^{(l)}_{I'}}(\hat{\mathbf{x}}_{jk})\hat{\psi}_{j^{(l)}_I k^{(l)}_I}(\hat{\mathbf{x}}_{jk}) \\
&= \sum_{l\in\mathcal{L}''}\delta_{j^{(l)}_I j^{(l)}_{I'}}\delta_{k^{(l)}_I k^{(l)}_{I'}}\alpha^{(l)}_{j^{(l)}_I k^{(l)}_I}\gamma(\mathbf{x}^{(l)}_{j^{(l)}_I k^{(l)}_I}) \\
&= \sum_{l\in\mathcal{L}''}\delta_{j^{(l)}_I j^{(l)}_{I'}}\delta_{k^{(l)}_I k^{(l)}_{I'}}a^r(l,j^{(l)}_I,k^{(l)}_I),
\end{aligned}$$

dove $a^r(i,j,k)=\alpha^{(i)}_{jk}\gamma(\mathbf{x}^{(i)}_{jk})$. e pertanto se $\mathcal{L}''=\varnothing$ vale

$$a^d_{I,I'}=0.$$

2.4.4 Costruzione della matrice di bordo

Per ogni funzione di base ψ_I, $\psi_{I'}$ indichiamo con $\mathcal{L}_D$ ($\mathcal{L}_N$ o $\mathcal{L}_R$) e $\mathcal{L}'_D$ ($\mathcal{L}'_N$ o $\mathcal{L}'_R$) gli insiemi degli indici relativi ai sottodomini per cui un lato di bordo di Dirichlet (Neumann o Robin) contenga il nodo $\mathbf{x}_I$, $\mathbf{x}'_I$ rispettivamente e con $\mathcal{L}''_D$ ($\mathcal{L}''_N$ o $\mathcal{L}''_R$) la loro intersezione.

Allora esistono $\{j^{(l)}_I\}_{l\in\mathcal{L}_D(\mathcal{L}_N,\mathcal{L}_R)}$, $\{k^{(l)}_I\}_{l\in\mathcal{L}_D(\mathcal{L}_N,\mathcal{L}_R)}$ tali per cui

$$\psi_{I\,|\Omega_l} = \hat{\psi}_{j^{(l)}_I k^{(l)}_I}\circ\hat{\boldsymbol{\phi}}^{(l)}, \qquad\qquad \forall l\in\mathcal{L}_D(\mathcal{L}_N,\mathcal{L}_R).$$

Sia poi $\mathcal{I}_i$ un'indiciazione dei nodi sul bordo Γ_i del dominio Ω. Allora la componente sulla I-esima riga della I'-esima colonna della matrice A^b relativa al termine forzante si scrive:

$$\begin{aligned}
a^b_{II'} &= \sum_{i\in\mathcal{I}_D} \eta\langle\!\langle \psi_{I'}, \psi_I\rangle\!\rangle_i + \sum_{i\in\mathcal{I}_R} \langle\!\langle \mu\gamma_i\psi_{I'}, \psi_I\rangle\!\rangle_i \\
&= \sum_{i\in\mathcal{I}_D} \eta \sum_{\mathbf{x}^{(i)}_{jk}\in\Gamma_i} \alpha^{(i)}_{jk}\psi_{I'}(\mathbf{x}^{(i)}_{jk})\psi_I(\mathbf{x}^{(i)}_{jk}) + \\
&\quad + \sum_{i\in\mathcal{I}_R} \sum_{\mathbf{x}^{(i)}_{jk}\in\Gamma_i} \alpha^{(i)}_{jk}\mu(\mathbf{x}^{(i)}_{jk})\gamma_i(\mathbf{x}^{(i)}_{jk})\psi_{I'}(\mathbf{x}^{(i)}_{jk})\psi_I(\mathbf{x}^{(i)}_{jk}) \\
&= \sum_{l\in\mathcal{L}''_D} \eta \sum_{\mathbf{x}^{(l)}_{jk}\in\Gamma_l} \alpha^{(l)}_{jk}\hat{\psi}_{j^{(l)}_{I'}k^{(l)}_{I'}}(\hat{\mathbf{x}}_{jk})\hat{\psi}_{j^{(l)}_{I}k^{(l)}_{I}}(\hat{\mathbf{x}}_{jk}) + \\
&\quad + \sum_{l\in\mathcal{L}''_R} \sum_{\mathbf{x}^{(l)}_{jk}\in\Gamma_l} \alpha^{(l)}_{jk}\mu(\mathbf{x}^{(l)}_{jk})\gamma_l(\mathbf{x}^{(l)}_{jk})\hat{\psi}_{j^{(l)}_{I'}k^{(l)}_{I'}}(\hat{\mathbf{x}}_{jk})\hat{\psi}_{j^{(l)}_{I}k^{(l)}_{I}}(\hat{\mathbf{x}}_{jk}) \\
&= \sum_{l\in\mathcal{L}''_D} \delta_{j^{(l)}_I j^{(l)}_{I'}}\delta_{k^{(l)}_I k^{(l)}_{I'}}\eta\alpha^{(l)}_{j^{(l)}_I k^{(l)}_I} + \\
&\quad + \sum_{l\in\mathcal{L}''_R} \delta_{j^{(l)}_I j^{(l)}_{I'}}\delta_{k^{(l)}_I k^{(l)}_{I'}}\alpha^{(l)}_{j^{(l)}_I k^{(l)}_I}\mu(\mathbf{x}^{(l)}_{j^{(l)}_I k^{(l)}_I})\gamma_l(\mathbf{x}^{(l)}_{j^{(l)}_I k^{(l)}_I}) \\
&= \sum_{l\in\mathcal{L}''_D} \delta_{j^{(l)}_I j^{(l)}_{I'}}\delta_{k^{(l)}_I k^{(l)}_{I'}} a^b_D(l, j^{(l)}_I, k^{(l)}_I) + \sum_{l\in\mathcal{L}''_R} \delta_{j^{(l)}_I j^{(l)}_{I'}}\delta_{k^{(l)}_I k^{(l)}_{I'}} a^b_R(l, j^{(l)}_I, k^{(l)}_I),
\end{aligned}$$

dove $a^b_D(i,j,k) = \eta\alpha^{(i)}_{jk}$ ed $a^b_R(i,j,k) = \alpha^{(i)}_{jk}\mu(\mathbf{x}^{(i)}_{jk})\gamma_i(\mathbf{x}^{(i)}_{jk})$.

2.4.5 Costruzione del vettore forzante

La componente I-esima del vettore $\mathbf{f}$ relativo al termine forzante si scrive:

$$\begin{aligned}
f_I &= \langle f, \psi_I\rangle \\
&= \sum_{i}^{M}{}_{1} \sum_{jk}^{N}{}_{0} \alpha^{(i)}_{jk} f(\mathbf{x}^{(i)}_{jk})\psi_I(\mathbf{x}^{(i)}_{jk}) \\
&= \sum_{l\in\mathcal{L}} \sum_{jk}^{N}{}_{0} \alpha^{(l)}_{jk} f(\mathbf{x}^{(l)}_{jk})\psi^{(l)}_{j^{(l)}_I k^{(l)}_I}(\mathbf{x}^{(l)}_{jk}) \\
&= \sum_{l\in\mathcal{L}} \alpha^{(l)}_{j^{(l)}_I k^{(l)}_I} f(\mathbf{x}^{(l)}_{j^{(l)}_I k^{(l)}_I}).
\end{aligned}$$

2.4.6 Costruzione del vettore di bordo

La componente I-esima del vettore $\mathbf{f}^b$ relativo al termine di bordo si scrive:

$$\begin{aligned}
f_I^b &= \sum_{i\in\mathcal{I}_N} \langle\!\langle \mu h_i, \psi_I \rangle\!\rangle_i + \sum_{i\in\mathcal{I}_R} \langle\!\langle \mu r_i, \psi_I \rangle\!\rangle_i \\
&= \sum_{i\in\mathcal{I}_N} \sum_{\mathbf{x}_{jk}^{(i)}\in\Gamma_i} \alpha_{jk}^{(i)} \mu(\mathbf{x}_{jk}^{(i)}) h_i(\mathbf{x}_{jk}^{(i)}) \psi_I(\mathbf{x}_{jk}^{(i)}) + \\
&\quad + \sum_{i\in\mathcal{I}_R} \sum_{\mathbf{x}_{jk}^{(i)}\in\Gamma_i} \alpha_{jk}^{(i)} \mu(\mathbf{x}_{jk}^{(i)}) r_i(\mathbf{x}_{jk}^{(i)}) \psi_I(\mathbf{x}_{jk}^{(i)}) \\
&= \sum_{l\in\mathcal{L}_N} \sum_{\mathbf{x}_{jk}^{(l)}\in\Gamma_l} \alpha_{jk}^{(l)} \mu(\mathbf{x}_{jk}^{(l)}) h_l(\mathbf{x}_{jk}^{(l)}) [\hat{\psi}_{j_I^{(l)} k_I^{(l)}} \circ \hat{\boldsymbol{\phi}}^{(l)}(\mathbf{x}_{jk}^{(l)})] + \\
&\quad + \sum_{l\in\mathcal{L}_R} \sum_{\mathbf{x}_{jk}^{(l)}\in\Gamma_l} \alpha_{jk}^{(l)} \mu(\mathbf{x}_{jk}^{(l)}) r_l(\mathbf{x}_{jk}^{(l)}) [\hat{\psi}_{j_I^{(l)} k_I^{(l)}} \circ \hat{\boldsymbol{\phi}}^{(l)}(\mathbf{x}_{jk}^{(l)})] \\
&= \sum_{l\in\mathcal{L}_N} \sum_{\mathbf{x}_{jk}^{(l)}\in\Gamma_l} \alpha_{jk}^{(l)} \mu(\mathbf{x}_{jk}^{(l)}) h_l(\mathbf{x}_{jk}^{(l)}) \hat{\psi}_{j_I^{(l)} k_I^{(l)}}(\hat{\mathbf{x}}_{jk}) + \\
&\quad + \sum_{l\in\mathcal{L}_R} \sum_{\mathbf{x}_{jk}^{(l)}\in\Gamma_l} \alpha_{jk}^{(l)} \mu(\mathbf{x}_{jk}^{(l)}) r_l(\mathbf{x}_{jk}^{(l)}) \hat{\psi}_{j_I^{(l)} k_I^{(l)}}(\hat{\mathbf{x}}_{jk}) \\
&= \sum_{l\in\mathcal{L}_N} \alpha_{j_I^{(l)} k_I^{(l)}}^{(l)} \mu(\mathbf{x}_{j_I^{(l)} k_I^{(l)}}^{(l)}) h_l(\mathbf{x}_{j_I^{(l)} k_I^{(l)}}^{(l)}) + \\
&\quad + \sum_{l\in\mathcal{L}_R} \alpha_{j_I^{(l)} k_I^{(l)}}^{(l)} \mu(\mathbf{x}_{j_I^{(l)} k_I^{(l)}}^{(l)}) r_l(\mathbf{x}_{j_I^{(l)} k_I^{(l)}}^{(l)}).
\end{aligned}$$

2.5 Solver

Per la fase di solving abbiamo impiegato gli algoritmi implementati nella libreria `Jama` sviluppata al NIST. Questa fornisce, oltre alle routine per la decomposizione di una matrice ai valori singolari e agli autovalori, i seguenti metodi per la risoluzione di un sistema algebrico nella forma $A\mathbf{x} = \mathbf{b}$ con A matrice quadrata di ordine n

Decomposizione di Cholesky calcola, cioè, una matrice triangolare inferiore L tale che $A = LL^T$. Una tale matrice esiste se A è una matrice simmetrica definita positiva, in questo caso si può ricavare la soluzione $\mathbf{x}$ risolvendo in successione $L\mathbf{y} = \mathbf{b}$ e poi $L^T\mathbf{x} = y$. In genere la complessità computazionale degli algoritmi usati comunemente è $O(n^3)$ e richiedono circa $\frac{1}{3}n^3$ FLOP[12], pertanto il costo computazionale dell'algoritmo di Cholesky è circa la metà della decomposizione LU che richiede circa $\frac{2}{3}n^3$ FLOP.

LU con pivoting parziale calcola, cioè una matrice triangolare inferiore L, una triangolare superiore U ed una matrice di permutazione P tali che $PA = LU$. Una tale decomposizione esiste sempre, ma può essere utilizzata per ricavare la soluzione $\mathbf{x}$ solo se A non è singolare, in questo caso si risolvono in successione $L\mathbf{y} = \mathbf{b}$ e poi $U\mathbf{x} = y$. In genere la complessità

[12]Numero di operazioni in virgola mobile (Floating point operations)

computazionale degli algoritmi usati comunemente è $O(n^3)$ e richiedono circa $\frac{2}{3}n^3$ FLOP; il pivoting parziale vi somma solamente un termine quadratico.

QR calcola, cioè una matrice ortogonale Q ed una triangolare superiore R tali che $A = QR$. Una tale decomposizione esiste sempre, ma può essere utilizzata per ricavare la soluzione $\mathbf{x}$ solo se A è di rango pieno, in questo caso si minimizza la quantità $\|QR\mathbf{x} - \mathbf{b}\|_2$, ovvero $\mathbf{x}$ è la soluzione ai minimi quadrati del problema. In genere la complessità computazionale degli algoritmi usati comunemente è $O(n^3)$ e richiedono, nel caso delle trasformazioni di Householder, $\frac{2}{3}n^3 + n^2 + \frac{1}{3}n - 2$ FLOP.

2.5.1 Algoritmo di Cholesky

Poniamo

$$A =: A^{(1)} = \left(\begin{array}{c|c} a_{1,1}^{(1)} & \mathbf{b}^{(1)^T} \\ \hline \mathbf{b}^{(1)} & B^{(1)} \end{array}\right)$$

e, per ogni $i = 1, \ldots, n$, definiamo

$$L^{(i)} = \left(\begin{array}{c|c|c} \mathbb{I}_{i-1} & & \\ \hline & \sqrt{a_{i,i}^{(i)}} & \\ \hline & \frac{1}{\sqrt{a_{i,i}^{(i)}}}\mathbf{b}^{(i)} & \mathbb{I}_{n-1} \end{array}\right)$$

e definiamo implicitamente $A^{(i+1)}$ tramite

$$A^{(i)} = L^{(i)} A^{(i+1)} {L^{(i)}}^T;$$

allora :

$$A^{(i+1)} := \left(\begin{array}{c|c|c} \mathbb{I}_{i-1} & & \\ \hline & 1 & \\ \hline & & B^{(i)} - \frac{1}{\sqrt{a_{i,i}^{(i)}}}\mathbf{b}^{(i)}\mathbf{b}^{(i)^T} \end{array}\right)$$

$$= \left(\begin{array}{c|c|c} \mathbb{I}_i & & \\ \hline & a_{i+1,i+1}^{(i+1)} & \mathbf{b}^{(i+1)^T} \\ \hline & \mathbf{b}^{(i+1)} & B^{(i+1)} \end{array}\right)$$

Si ottiene dopo n passi $A^{(n+1)} = \mathbb{I}_n$, pertanto la matrice triangolare inferiore L può essere calcolata come $L = \prod_{i=1}^{n} L_i$.

2.5.2 Algoritmo di Doolitle per la fattorizzazione LU

Poniamo

$$A =: A^{(1)} = \left(\begin{array}{c|ccc} a_{1,1}^{(1)} & * & \cdots & * \\ \hline & * & \cdots & * \\ \mathbf{b}^{(i)} & \vdots & \ddots & \vdots \\ & * & \cdots & * \end{array}\right)$$

allora per ogni $i = 1, \ldots, n-1$ definiamo

$$L^{(i)} = \left(\begin{array}{c|c|c} \mathbb{I}_{i-1} & & \\ \hline & 1 & \\ \hline & \frac{1}{a_{i,i}^{(i)}}\mathbf{b}^{(i)} & \mathbb{I}_{n-1} \end{array}\right)$$

e calcoliamo

$$A^{(i+1)} := L^{(i)}A^{(i)} = \left(\begin{array}{ccc|c|ccc} * & \cdots & * & * & * & \cdots & * \\ & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ & & * & * & * & \cdots & * \\ \hline & & & a_{i+1,i+1}^{(i+1)} & * & \cdots & * \\ \hline & & & & * & \cdots & * \\ & & & \mathbf{b}^{(i+1)} & \vdots & \ddots & \vdots \\ & & & & * & \cdots & * \end{array}\right).$$

Si ottiene dopo $n-1$ passi $A^{(n)} =: U$ matrice triangolare superiore, pertanto la matrice triangolare inferiore L può essere calcolata come $L = \prod_{i=1}^{n-1} L^{(i)}$. L'algoritmo con pivoting parziale introduce ad ogni passo dell'iterazione una permutazione delle righe di $A^{(i)}$ in modo tale da garantire che l'elemento $a_{i,i}^{(i)}$ sia quello maggiore in modulo se esiste un elemento non nullo nella i-esima colonna, altrimenti, procede a fattorizzare la sottomatrice ottenuta eliminando la i-esima rigla e la i-esima colonna.

2.5.3 Algoritmo delle trasfomazioni di Housholder per la fattorizzazione QR

È una generalizzazione dell'algoritmo di Gram-Schmidt che riportiamo di seguito. Indichiamo con $\pi_{\mathbf{b}}\mathbf{a}$ la proiezione di $\mathbf{a}$ su $\mathbf{b}$, $\pi_{\mathbf{b}}\mathbf{a} = \frac{\mathbf{a}\cdot\mathbf{b}}{\mathbf{b}\cdot\mathbf{b}}\mathbf{b}$ e con $\mathbf{a}_i$ le colonne di A. Poniamo $\mathbf{u}_i = \mathbf{a}_1$ e definiamo per ogni $i = 1, \ldots, n$:

$$\mathbf{e}_i = \frac{\mathbf{u}_i}{|\mathbf{u}_i|},$$

$$\mathbf{u}_{i+1} = \mathbf{a}_{i+1} - \pi_{\mathbf{e}_i}\mathbf{a}_{i+1};$$

allora, se definiamo $Q = (\mathbf{e}_1, \cdots, \mathbf{e}_n)$ e

$$R = \begin{pmatrix} \mathbf{e}_1 \cdot \mathbf{a}_1 & \mathbf{e}_1 \cdot \mathbf{a}_2 & \cdots & \mathbf{e}_1 \cdot \mathbf{a}_n \\ & \mathbf{e}_1 \cdot \mathbf{a}_2 & \cdots & \mathbf{e}_1 \cdot \mathbf{a}_n \\ & & \ddots & \vdots \\ & & & \mathbf{e}_n \cdot \mathbf{a}_n \end{pmatrix},$$

otteniamo la decomposizione desiderata $A = QR$.

La fattorizzazione tramite le trasformazioni di Householder si ottiene utilizzando queste trasformazioni al posto della proiezione ortogonale π. In particolare si usa $\pi^H_{\mathbf{b}}\mathbf{a} := \frac{|\mathbf{a}|}{|\mathbf{e}|}\mathbf{e}$.

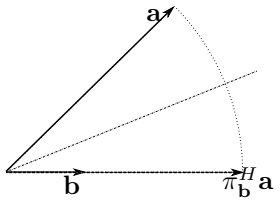

Figura 2.3: Trasformazione di Householder per la fattorizzazione QR

2.6 PostProcessor

Nel namespace `SemSolver::PostProcessor` sono implementate poche funzioni che elaborano il risultato algebrico fornito dal solutore. Queste sono meramente funzionali alla visualizzazione grafica del risultato. In questo namespace si potrebbero implementare ulteriori routine che permettano, ad esempio, di valutare l'errore compiuto dal modello nella risoluzione di un problema la cui soluzione sia analiticamente nota.

2.7 I/O

Nel namespace `SemSolver:IO` sono implementate alcune funzioni che implementano le funzionalità di input/output su file. In particolare sono definiti, oltre al Triangle PSLG (`.poly`), i seguenti tipi di file

SemSolver Workspace (`.semsol`) È un archivio di tipo `tar`[13] all'interno del quale possono essere raggruppati più file relativi ad uno o più problemi. La GUI del programma permette di lavorare su un solo file Workspace alla volta.

SemSolver Geometry (`.semgeo`) È anch'esso un `tarball`. Affinchè sia utilibizzabile dal programma deve contenere al suo interno un file denominato `pslg.poly` di tipo Triangle PSLG (`.poly`) che descriva un determinato dominio Ω ed un file denominato `domains.semsub` che ne descriva una

[13]Il formato `tar` è stato reso uno standard da POSIX.1-1998 e dal successivo POSIX.1-2001. Offre possibilità di accumulare dentro un singolo file migliaia di file diversi (packing). `tar` non supporta direttamente la compressione.

partizione quadrangolare. Il file `domains.semsub` è così strutturato: nella prima riga vi è indicato il numero di elementi della partizione e successivamente per ogni elemento vi è una sezione costituita da una prima riga, indicante l'indice (non negativo) dell'elemento ed il numero di vertici che lo definiscono, e da tante righe quanti sono i vertici, costituite dalle coordinate di ciascun vertice seguite da un numero.; se questo numero è non negativo indica l'indice corrispondente all'elemento con in comune il lato che finisce in quel nodo, altrimenti, indica l'opposto diminuito di uno dell'indice relativo al lato del dominio di cui il lato che finisce in quel nodo è parte. Ad esempio il file `rectangle.semgeo` che descrive la seguente geometria

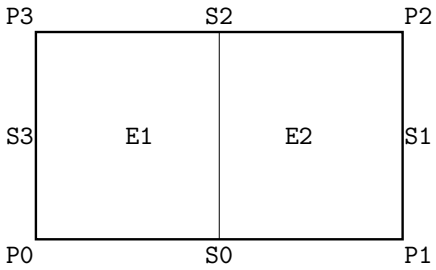

Figura 2.4: Esempio di geometria

conterrà il seguente file `pslg.poly`

```
#
# veritces
#
# num    dim    attr    mark
   4      2       0       0
# id    x    y
   0    0    0
   1    2    0
   2    2    1
   3    0    1
#
# segments
#
   4     0
# id    from    to
   0      1      2
   1      2      3
   2      3      4
   3      4      1
#
# holes
#
# num
   0
```

ed il seguente file `domains.semsub`

```
# number of elements
   2

# first element
   1   4
   0   0   -4
   1   0   -1
   1   1    2
   0   1   -3
```

```
# second element
   2   4
   1   0    1
   2   0   -1
   2   1   -2
   1   1   -3
```

SemSolver Boundary Conditions (`.sembcs`) È un file di testo costituito da una riga per ogni condizione al bordo, costituita ciascuna dall'etichetta relativa la bordo su cui imporre la condizione, dal tipo della condizione (`DIRICHLET`, `NEUMANN` o `ROBIN`) e dalla definizione dei relativi dati in ECMA Script. Ad esempio se vogliamo definire sulla geometria precedente le seguenti condizioni al bordo

$$\begin{cases} u = \sin(x)\cos(y) & \texttt{su } \Gamma_0 \cup \Gamma_2 \\ \nabla u \cdot \mathbf{n} = 0 & \texttt{su } \Gamma_1 \\ \nabla u \cdot \mathbf{n} + \frac{x}{7} 3u = \cos(x) & \texttt{su } \Gamma_1 \end{cases}$$

useremo il seguente file `miste.sembcs`

```
   0   DIRICHLET   sin(x)*cos(x)
   1   NEUMANN                0
   2   DIRICHLET   sin(x)*cos(x)
   3   ROBIN               x/3   cos(x)
```

SemSolver Equation (`.semeqn`) È un file di testo costituito da una prima riga che ne indica il tipo dell'equazione (`DIFFUSION_CONVECTION_REACTION`) seguita da una una riga per ogni parametro (`DIFFUSION`, `CONVECTION`, `REACTION` e `FORCING`) dell'equazione. Ad esempio qualora volessimo risolvere la seguente equazione di Poisson

$$\nabla^2 u(x, y) = \mathrm{e}^x$$

utilizzeremmo il file `poisson.semeqn`

```
DIFFUSION_CONVECTION_REACTION
DIFFUSION       1
CONVECTION      0       0
REACTION        0
FORCING         exp(x)
```

SemSolver Paramters (`.semprm`) È un file di testo costituito da tre righe: una per indicar il grado degli elementi spettrali da utilizzare (`DEGREE`), una per la tolleranza (`TOLERANCE`) ed una per il coefficiente di penalità η (`PENALITY`). Ad esempio qualora volessimo impiegare elementi spettrali quadratici potremmo utilizzare il file `quadratic.semprm`

```
DEGREE          2
TOLERANCE       1e-06
PENALITY        1e+06
```

2.8 GUI

Per la realizzazione dell'interfaccia grafica abbiamo deciso di ricorrere alle librerie Qt perchè garantiscono la portabilità sulle principali piattaforme, infatti permettono di definire GUI sia in ambilente Linux, Mac e Windows.

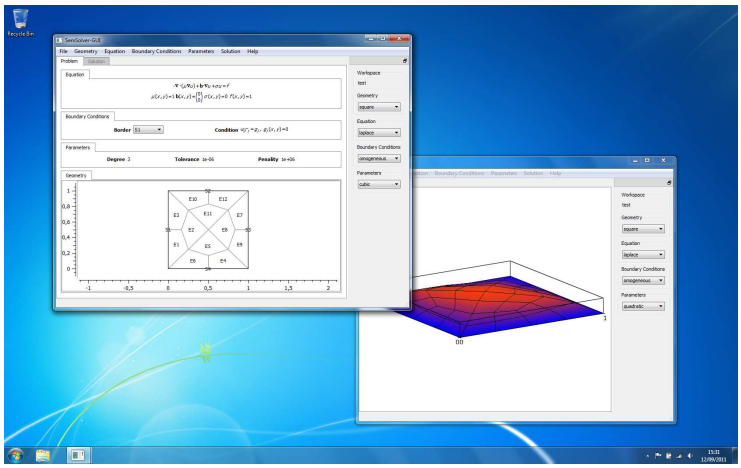

Figura 2.5: Il programma `SemSolver-GUI` in ambiente Windows (Windows 7)

Volevamo che attravero la GUI fosse possibile, oltre alla visualizzazione dei risultati, anche la definizione degli input e la loro visualizzazione.
La GUI è costituita da una finestra principale: un pannello mostra il contenuto del workspace corrente e due tab permettono la visualizzazione degli input selezionati e dell'output calcolato.

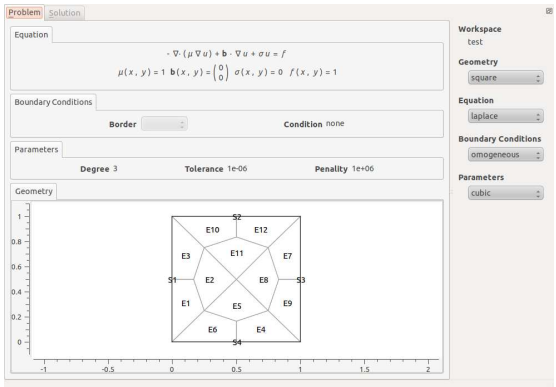

Figura 2.6: La finestra principale

Tramite il menù a barra è inoltre possibile accedere alle finestre per l'apertura/salvataggio dei workspace, alle finestre tramite le quali creare, importare, esportare o rimuovere un file di input nel workspace corrente.

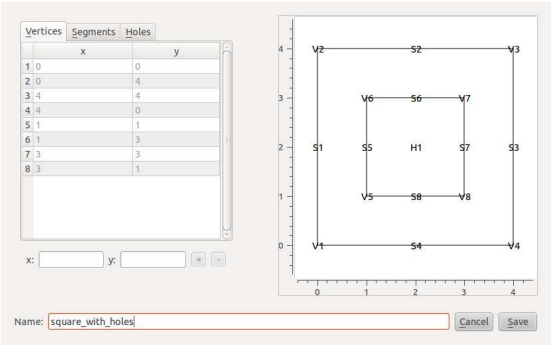

Figura 2.7: La finestra per creare una nuova geometria

Sempre dal menù a barra si può salvare la soluzione calcolata in un file testuale ed in vari formati grafici.

Capitolo 3

Analisi dati

Presentiamo ora un'analisi della qualità del metodo implementato nel programma. Abbiamo analizzato

- l'andamento dell'errore all'aumentare del grado N degli elementi spettrali;
- l'andamento dell'errore all'aumentare del numero M di sottodomini computazionali;
- la stabilità della soluzione ottenuta all'aumentare del numero di Péclet globale.

Per i primi due punti abbiamo considerato il seguente problema di Poisson con condizioni al contorno di Dirichlet omogenee

$$\begin{cases} -\nabla^2 u(\mathbf{x}) = 1 & \forall \mathbf{x} \in \Omega \\ u(\mathbf{x}) = 0 & \forall \mathbf{x} \in \partial\Omega \end{cases}$$

Come soluzione esatta abbiamo utilizzato la soluzione ottenuta con il metodo degli elementi finiti su una griglia fitta tramite il tool `PDEtool` di `Matlab` e

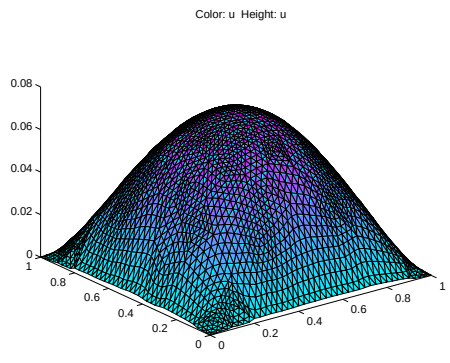

Figura 3.1: La soluzione ottenuta con PDETool

abbiamo successivamente stimato, sempre in Matlab, l'errore $\|u - u_N\|_L^2$ compiuto nel calcolare le soluzioni per $N = 2, 3, 4, 8$ sulle seguenti geometrie per $M = 1, 2, 4, 12$.

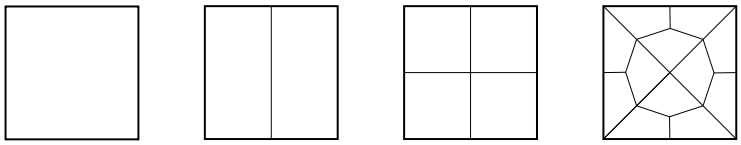

Figura 3.2: La geometrie considerate

Gli errori ottenuti sono illustrati nei seguenti grafici in scala logaritmica

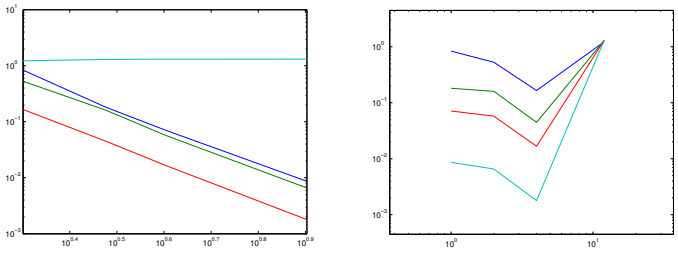

Figura 3.3: L'andamento dell'errore all'aumentare di N (a sinistra) ed all'aumentare di M (a destra).

Dai grafici si può osservare come l'errore diminuisca quadraticamente all'aumentare del grado degli elementi spettrali quando si usano le geometrie con $M = 1, 2, 4$ mentre risulta costante per $M = 12$. Tale comportamento va imputato, probabilmente, al fatto che quest'ultima geometria presenta elementi non "regolari" a differenza delle altre e pertanto si introduce un errore dovuto alle approssimazioni nel calcolo delle trasformazioni $\boldsymbol{\phi}_k$.

Possiamo inoltre riscontrare come l'andamento dell'errore sia lineare all'aumentare del numero di sottodomini computazionali fintanto che si considerino sottodomini "regolari".

Per quanto riguarda l'analisi della stabilità numerica del metodo implementato, abbiamo considerato il seguente problema di diffusione-trasporto:

$$\begin{cases} -\mu\nabla^2 u(\mathbf{x}) + \nabla \cdot u(\mathbf{x}) = 1 & \forall \mathbf{x} \in \Omega \\ u(\mathbf{x}) = 0 & \forall \mathbf{x} \in \partial\Omega \end{cases}$$

al variare di $\mu \in \mathbb{R}$ con elementi di grado $N = 4$ sulla geometria precedente per $M = 4$. Si osserva che già per valori di μ dell'ordine di 10^{-3} la soluzione presenta instabilità numeriche.

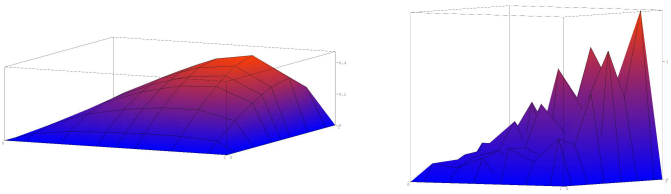

Figura 3.4: L'andamento dell'errore all'aumentare di N (a sinistra) ed all'aumentare di M (a destra).

Tale comportamento è dovuto al fatto che nel metodo implementato non è stat introdotta alcuna stabilizzazione per la risoluzione del problema algebrico.

Capitolo 4

Conclusioni

Il progetto che abbiamo presentato consente la risoluzione di problemi ellittici generali tramite il metodo degli elementi spettrali con integrazione numerica. Si è visto come questo programma riesca a creare, a partire da una geometria poligonale qualsiasi, una quadrangolazione del dominio opportuna ed a risolvervi un'equazione di diffusione-trasporto-reazione definita in run-time.

Nell'implementazione del programma si è prestata attenzione anche alla realizzazione di una interfaccia grafica per permettere l'utilizzo intuitivo del programma sviluppato e la gestione di file di input/output di semplice codifica in modo che fosse possibile specificare manualmente anche input particolari che l'interfaccia non è in grado di gestire da sola (ad esempio griglie di calcolo particolari).

Nella scrittura del codice si è cercato di garantire la leggibilità del listato e la possibilità di implementare generalizzazioni ed estensioni delle funzionalità finora implementate, questo grazie all'uso di classi template e virtuali.

Il codice si è dimostrato stabile solo su geometrie e problemi "regolari". Per ovviare a tali problemi sarebbe necessario implementare tecniche di precondizionamento e stabilizzazione del problema algebrico ed introdurre tipologie ulteriori di trasformazioni dei sottodomini nel dominio di riferimento.