Metodi numerici per le PDE — Relazione sul metodo degli elementi finiti in MATLAB

Lilia Grasso, matricola 813210
Davide Iavarone, matricola 784697

20/06/2020

1 Esercizio 1

La risoluzione dell'esercizio si suddivide in quattro parti:

1. Scegliere varie famiglie di mesh con diametro massimo dei triangoli che tende a zero e riportare su un grafico in scala logaritmica h e gli errori corrispondenti. Verificare che le pendenze (cioè gli ordini) siano quelli previsti caso per caso ($k = 1, 2, 3$ errore in $L^2(\Omega)$, errore in $H^1(\Omega)$, $h = h_{medio}, h = h_{max}$).
2. Verificare che la dimensione del sistema lineare è circa k^2 volte il numero di vertici.
3. Verificare che se si utilizza una formula di quadratura di grado di precisione maggiore di $2k - 1$ gli ordini di convergenza non cambiano.
4. Verificare che se la soluzione esatta è un polinomio di grado k gli errori invece dipendono dalla formula di quadratura utilizzata.

1.1 Introduzione ai codici

Per la risoluzione dell'esercizio, come dominio abbiamo considerato un quadrato di lato $l = 1$ e di coordinate$A_1 = (0, 0)$, $B_1 = (1, 0)$, $C_1 = (1, 1)$, $D_1 = (0, 1)$, e abbiamo usato due tipi di famiglie di mesh con diametro massimo dei triangoli che "tende a zero" annidate, quelle strutturate (ndr 'quadrato mesh strutturata'), e quelle non strutturate (ndr 'quadrato es1'). I coefficienti di reazione e trasporto sono stati scelti in modo tale che rispettassero la condizione $\alpha(x, y) - \frac{1}{2} div\beta(x, y) \geq 0$ e $c(x, y) \geq c_0 > 0$ con $(x, y) \in \Omega$.

Abbiamo definito le seguenti funzioni:

1. $\alpha = alfa$ coefficiente di reazione come $alfa(x, y) = x^2 + y^2$;
2. $\beta = beta1$ coefficiente di trasporto come $beta1(x, y) = \binom{-y}{-x}$;
3. c coefficiente di diffusione come $c2d(x, y) = x^2 + y^2 + 1$;
4. f termine noto che varia in base al k considerato nel caso in cui come u esatta scegliamo un polinomio di grado k mentre nel caso in cui si sceglie una soluzione esatta non polinomiale è uguale in tutti e tre i casi;
5. $g = gD$ valore di u al bordo uguale a $uesattanonpolinomio$;
6. $uesattanonpolinomio$ come $uesatta(x, y) = sin(\pi x)sin(\pi y) + x^2 + y^2 + 2$;
7. $graduesatta(x, y) = \binom{\pi cos(\pi x)sin(\pi y)+2x}{\pi cos(\pi y)sin(\pi y)+2y}$.

Il problema da risolvere è studiare l'ordine di convergenza in $L^2(\Omega)$ e in $H^1(\Omega)$ del metodo degli elementi finiti in due dimensioni basato sui triangoli e sui polinomi di grado 1, 2 e 3 che approssimi la soluzione dell'equazione differenziale alle derivate parziali:

$$\begin{cases} -div(c\nabla u) + \beta\nabla u + \alpha u = f & in\ \ \Omega \\ u = g & su\ \ \Gamma = \partial\Omega \end{cases}$$

il quale corrisponde al seguente problema variazionale, di trovare $u \in H^1_{\Gamma,g}(\Omega)$ tale che

$$\int_\Omega c(\nabla u \cdot \nabla v)\, dx + \int_\Omega (\beta \cdot \nabla u)v\, dx + \int_\Omega \alpha uv\, dx = \int_\Omega fv\, dx \quad \forall v \in H^1_{\Gamma,0}(\Omega).$$

Si ha come spazio di approssimazione finito dimensionale $V^k_h \subset H^1_{\Gamma,0}(\Omega)$,il quale è l'insieme delle v_h continue in Ω e nulle sul bordo di Ω, le quali ristrette al triangolo T sono un polinomio di grado k. Presa $u_h \in V^k_{h,g} = \{v_h : \Omega \longrightarrow \mathbb{R}$ t.c $v_h\, su\, T$ è un polinomio di grado k, $v_h \in C^0(\bar{\Omega})$, e $v_h = g$ in $\partial\Omega\}$, si ha, prendendo la base di lagrange $\{\varphi_i\}_{i=1}^{N_k}$ con N dimensione dello spazio dei polinomi $\mathbb{P}_k(\mathbb{R}^2)$:

$$u_h = \sum_{i=1}^{N_k} u_i\varphi_i,$$

dove gli u_i sono proprio le componenti del vettore soluzione $\vec{u_h}$ nei nodi dell'elemento T. Per calcolare gli integrali ci si riporta a integrali sui triangoli:

$$\int_\Omega c(\nabla u_h \cdot \nabla v_h)\,dx + \int_\Omega (\beta \cdot \nabla u_h) v\,dx + \int_\Omega \tilde{\alpha} u_h v_h\,dx =$$
$$= \sum_T \left\{ \int_T c(\nabla u_h \cdot \nabla v_h)\,dx + \int_T (\beta \cdot \nabla u_h) v_h\,dx + \int_T \tilde{\alpha} u_h v_h\,dx \right\} =$$
$$\sum_T \int_T f v_h\,dx \quad \forall v_h \in V^k_{h,0}.$$

Si ottiene in questo modo un sistema lineare da risolvere:

$$(A + A_{tras} + A_{reaz})\vec{u_h} = \vec{f_h}.$$

Questi integrali possono essere calcolati tramite formule di quadratura sul triangolo di riferimento $\hat{T}$ di vertici $\hat{v_1} = (0,0)$, $\hat{v_2} = (0,1)$, $\hat{v_3} = (0,1)$ usando $F_T(\hat{x}) = J_{F_T}\hat{x} + \vec{x_0} = x$, la mappa affine che manda gli elementi del triangolo elementare nel generico triangolo della triangolazione, dove J_{F_T} è lo jacobiano della trasformazione $J_{F_T} = \begin{Bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{Bmatrix}$.

Gli errori in $L^2(\Omega)$ e in $H^1(\Omega)$ rispetto alla soluzione esatta u_{ex} sono definiti da:

$$errL2 = \left(\int_\Omega (u_{ex}(x) - u_h(x))^2\,dx \right)^{\frac{1}{2}}$$

$$errH1 = \left(\int_\Omega |\ \nabla u_{ex}(x) - \nabla u_h(x)\ |^2\ dx \right)^{\frac{1}{2}}.$$

I codici sono stati suddivisi per il grado degli elementi finiti considerati, dove per ognuno abbiamo identificato con ne il numero di nodi per triangolo considerato, e inizializzato un ciclo for che prende 5 mesh annidate generate precedentemente da $refine$. Abbiamo calcolato le richieste all'interno dei seguenti codici:

1. MainP1 per gli elementi finiti di grado 1, con $ne = 3$;
2. MainP2 per gli elementi finiti di grado 2 con $ne = 6$;
3. MainP3 per gli elementi finiti di grado 3 con $ne = 10$;

All'interno di ogni main son state definite le funzioni di base le $\hat{\varphi_j}(\hat{x})$ del triangolo elementare che sono le seguenti per esempio nel caso $k = 1$:

$$phi1 = \begin{cases} \hat{\varphi_1}(\hat{x}, \hat{y}) = 1 - \hat{x} - \hat{y} \\ \hat{\varphi_2}(\hat{x}, \hat{y}) = \hat{x} \\ \hat{\varphi_3}(\hat{x}, \hat{y}) = \hat{y} \end{cases}.$$

Nel main inoltre per ogni mesh è stata calcolata la soluzione approssimata u_h, attraverso le seguenti funzioni:

1. Lettura della mesh: attraverso $readmesh, readedges$ e $preprocessing.mesh$, il programma legge il file .node, .ele e .edge generato da Triangle.
2. Calcolo degli integrali: attraverso $femPkes1$ il programma calcola la matrice e il termine noto per ogni triangolo, andando poi a spalmare i valori all'interno della matrice globale A. Per calcolare $\nabla\varphi_j$ abbiamo utilizzato la mappa affine F_T che calcola ogni integrale sul triangolo di riferimento. Abbiamo calcolato $\hat{\varphi_i}$, $\hat{\nabla}\hat{\varphi_i}$ all'inizio di ogni main.
3. Assegnazione dei nodi con condizioni al bordo di Dirichlet: attraverso la funzione $nodi.Dirichlet$, il programma assegna i nodi di Dirichlet e li inserisce all'interno della matrice A e nel termine noto, $f = b = gD(xnodo, ynodo)$. Attraverso la funzione $sistema.effettivo$, si estrae il sistema effettivo dopo aver eliminato le colonne relative ai nodi con condizioni al bordo di Dirichlet da A e f, dando in output Ae, fe.

4. Soluzione sistema lineare: dopo aver inizializzato u_h la calcoliamo sui nodi effettivi assegnandogli il valore Ae/fe, e il valore di gD nei nodi di bordo.

1.1.1 Implementazioni condizioni al bordo di Dirichlet

Le condizioni al bordo nel nostro caso sono condizioni essenziali e nella formulazione variazionale sono inserite all'interno dello spazio in cui si cerca la soluzione u_h, ovvero $V^k_{h,g}$. Questo spazio è uno spazio affine in quanto è scrivibile come $V^k_{h,g} = G + V^k_{h,0}$ dove $V^k_{h,0} = \{v_h : \Omega \longrightarrow \mathbb{R}$ t.c $v_h \, su \, T$ è un polinomio di grado k, $v_h \in C^0(\bar{\Omega})$, e $v_h = 0$ in $\partial\Omega\}$. Definiti:

1. N_L nodi liberi, che sono unione dei nodi interni e i nodi di Neumann;
2. N_D non di Dirichlet;

si ha che $N_{tot} = N_L + N_D$. Allora $u_h = \sum_{j \in N_L} u_j \varphi_j + \sum_{\alpha \in N_D} g(x_\alpha)\varphi_\alpha$ e il problema sarà di trovare $u_h \in V^k_{h,g}$ tale che:

$$\int_\Omega c(\nabla u_h \cdot \nabla v_h)\, dx + \int_\Omega (\beta \cdot \nabla u_h) v_h \, dx + \int_\Omega \tilde{\alpha} u_h v_h \, dx = \int_\Omega f v_h \, dx \quad \forall v_h \in V^k_{h,0}.$$

Sostituendo il valore di u_h con quello menzionato sopra, e il valore di v_h con φ_i si ottiene:

$$\sum_{j \in N_L} \left(\int_\Omega c(\nabla \varphi_j \cdot \nabla \varphi_i)\, dx \right) u_j + \sum_{\alpha \in N_D} \left(\int_\Omega c(\nabla \varphi_\alpha \cdot \nabla \varphi_i)\, dx \right) g(x_\alpha) +$$

$$\sum_{j \in N_L} \left(\int_\Omega (\beta \cdot \nabla \varphi_j)\varphi_i \, dx \right) u_j + \sum_{\alpha \in N_D} \left(\int_\Omega (\beta \cdot \nabla \varphi_\alpha)\varphi_i \, dx \right) g(x_\alpha) +$$

$$\sum_{j \in N_L} \left(\int_\Omega \tilde{\alpha} \varphi_j \varphi_i \, dx \right) u_j + \sum_{\alpha \in N_D} \left(\int_\Omega \tilde{\alpha} \varphi_\alpha \varphi_i \, dx \right) g(x_\alpha) =$$

$$\int_\Omega f \varphi_i \, dx \quad \forall i \in N_D.$$

Considerando i valori delle sommatorie su $\alpha \in N_D$ come noti, si ottiene un nuovo sistema lineare con un differente termine noto:

$$(A + A_{tras} + A_{reaz})\vec{u_h} = \vec{f_h} + \vec{G_h}.$$

Per facilitare il calcolo di questo sistema abbiamo usato una logica particolare, ovvero abbiamo prima calcolato in $femPkes1$ il sistema come se tutti i nodi fossero liberi ottendendo la matrice $\hat{A}$ e il termine noto $\hat{f}$. La funzione *preprocessing mesh* richiamata per la lettura della mesh identifica tutti i nodi di Dirichlet (flag 1) e li salva nel vettore *indici nodi D*. Successivamente attraverso la funzione *nodi.Dirichlet* abbiamo imposto a posteriori delle condizioni al bordo di Dirichlet e identificato la colonna α-esima nella matrice A contenente i nodi di Dirichlet. Abbiamo successivamente sottratto al termine noto il valore $g(x_\alpha)$colonna α-esima; Infine con la funzione $sistema\, effettivo$ abbiamo eliminato la riga e la colonna α-esima dalla matrice $\hat{A}$ e dal termine noto $\hat{f}$, ottendendo così il sistema lineare con i tutti i nodi liberi su cui poi andremo a calcolare la u_h.

1.2 Calcolo convergenza in $L^2(\Omega)$ e in $H^1(\Omega)$

Dopo aver calcolato la u_h, abbiamo fatto l'analisi della convergenza attraverso le funzioni *calcolo errore*, *calcolo errore L2 Pk* e *calcolo errore H1 Pk* (per k=1 son stati calcolati esattamente i valori delle φ_i e $\nabla\varphi_i$ usando la formula di quadratura del baricentro in *calcolo errore L2 P1* e *calcolo errore H1 P1*).

Il calcolo della convergenza in $L2$ è stato fatto calcolando le $\hat{\varphi_j}(\hat{x})$ una volta per tutte prima del loop sugli elementi e considerando $(\hat{x_q}, \hat{\omega_q})$ con $q = 1, ..., N_q$, nodi e pesi di una formula di quadratura su $\hat{T}$ di riferimento.

1.2.1 Calcolo errore L2

Il codice *calcolo errore L2 Pk* prende in input le variabili *xnodo*,*ynodo*,*nodi elemento*,*fdq*,*ne*,*phi*,*uesatta*,*uh*, e tramite la funzione *quadratura* genera i nodi di quadratura in cui abbiamo calcolato le $\hat{\varphi}_j$. All'interno della funzione abbiamo un ciclo for per ogni elemento T (triangolo) dove viene eseguito il calcolo dell'errore in L2. In questo ciclo abbiamo calcolato prima i vettori dei lati $\vec{e_i}$ usando le coordinate dei vertici $x = xnodo, y = ynodo$ (calcolate nei nodi del triangolo considerato), e poi la matrice Jacobiana della trasformazione F_T che è la seguente: $J = \begin{Bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{Bmatrix}$.

Successivamente, dopo aver calcolaro i nodi di quadratura sull'elemento K, abbiamo calcolato la convergenza in L2 come segue:

$$(errL2)^2 = \int_\Omega \mid u_{ex}(x) - u_h(x) \mid^2 \, dx = \sum_T \int_T (u_{ex}(x) - u_h(x))^2 \, dx$$

e

$$\int_T \mid u_{ex}(x) - u_h(x) \mid^2 \, dx = \int_T (u_{ex}(x) - \sum_{j=1}^{N_k} u_j \varphi_j(x))^2 \, dx =$$

$$= 2 \mid T \mid \int_{\hat{T}} (u_{ex}(F_T(\hat{x}) - \sum_{j=1}^{N_k} u_j \hat{\varphi}_j(\hat{x}))^2 \, d\hat{x} =$$

$$\simeq 2 \mid T \mid \sum_{q=1}^{N_q} \left(u_{ex}(F_T(\hat{x_q}) - \sum_{j=1}^{N_k} u_j \hat{\varphi}_j(\hat{x_q}) \right)^2 \hat{\omega_q}.$$

1.2.2 Calcolo errore H1

Similmente al calcolo dell'errore in L2, per il calcolo della convergenza in H1 abbiamo implementato la funzione *calcolo errore H1 Pk* in cui abbiamo calcolato $\hat{\nabla}\hat{\varphi}_j$ invece delle $\hat{\varphi}_j$. Il calcolo dell'errore è stato approssimato con una formula di quadratura come segue:

$$(errH1)^2 = \int_\Omega \|\nabla u_{ex}(x) - \nabla u_h(x)\|_2^2 \, dx = \sum_T \int_T \|\nabla u_{ex}(x) - \nabla u_h(x)\|_2^2 \, dx$$

e

$$\int_T \|\nabla u_{ex}(x) - \nabla u_h(x)\|_2^2 \, dx = \int_T \|\nabla u_{ex}(x) - \sum_{j=1}^{N_k} u_i \nabla \varphi_j(x)\|_2^2 \, dx =$$

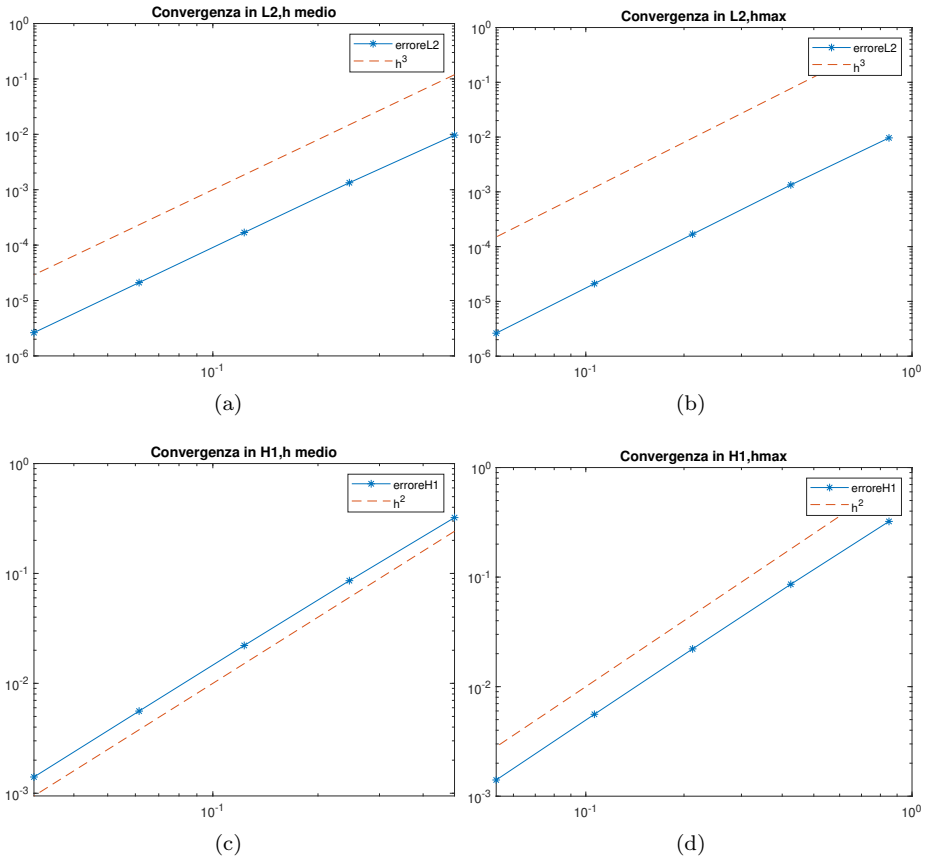
$$= 2 \mid T \mid \int_{\hat{T}} \|\nabla u_{ex}(F_T(\hat{x})) - \sum_{j=1}^{N_k} u_j J_{F_T}^{-T} \hat{\nabla}\hat{\varphi}_j(\hat{x})\|_2^2 \, d\hat{x} =$$

$$\simeq 2 \mid T \mid \sum_{q=1}^{N_q} \left\| \nabla u_{ex}(F_T(\hat{x_q})) - \sum_{j=1}^{N_k} u_j J_{F_T}^{-T} \hat{\nabla}\hat{\varphi}_j(\hat{x_q}) \right\|_2^2 \hat{\omega_q}.$$

A questo punto ci siamo serviti della formula

$$(A_1 - B_1)^2 + (A_2 - B_2)^2$$

dove A_1,A_2,B_1,B_2 sono le componenti rispettivamente dei vettori $A = \nabla u_{ex}(F_T(\hat{x_q}))$ e $B = \sum_{j=1}^{N_k} u_j J_{F_T}^{-T} \hat{\nabla}\hat{\varphi}_j(\hat{x_q})$.
Infine per ogni mesh abbiamo calcolato h_{mean}, che è la media dei diametri di ogni triangolo della mesh e *hmax* che è il diametro massimo di tutti i triangoli della mesh per raffinamento. Quindi abbiamo plottato i grafici logaritmici sia con ascissa h_{mean} che con ascissa *hmax* e con ordinata rispettivamente l'errore in L^2 e l'errore in H^1. Riportiamo di seguito i grafici nel caso $k = 2$ relativi all'errore H^1 e L^2 usando la soluzione esatta non polinomiale e usando come formula di quadratura $fdq = cubici$.



(a) (b) (c) (d)

Figure 1: Convergenza in H1 e L2 con fdq cubici

1.3 Calcolo dimensione del sistema lineare

Per verificare che la dimensione del sistema lineare è circa k^2V con V numero di vertici, abbiamo definito in ogni main la variabile $dim = size(A, 1)$ che da in output la dimensione della prima riga della matrice A. Infatti A è la matrice del sistema lineare associato agli integrali della formulazione variazionale e chiamando Π_k la dimensione di $\mathbb{P}_k(\mathbb{R}^2)$ si ha:

$$dim\mathbb{P}_k(\mathbb{R}^2) = \Pi_k$$

e che

$$A \in \mathbb{R}^{\Pi_k \times \Pi_k}.$$

Dunque avendo che il numero di vertici corrisponde al numero di vertici all'interno del vettore $xnodo$, abbiamo definito $V = lenght(xnodo)$ e moltiplicato V per 1 nel MainP1, per 2^2 nel MainP2 e infine per 3^2 nel MainP3 ottenendo così la dimensione di V_h^k. Definendo $dimR$ le dimensioni calcolate nei main con la formula k^2V, le abbiamo poi stampate entrambe col comando $disp([dim, dimR])$, per verificare che siano circa uguali.

1.4 Calcolo convergenza in $L^2(\Omega)$ e in $H^1(\Omega)$ con formula di quadratura maggiore di $2k-1$

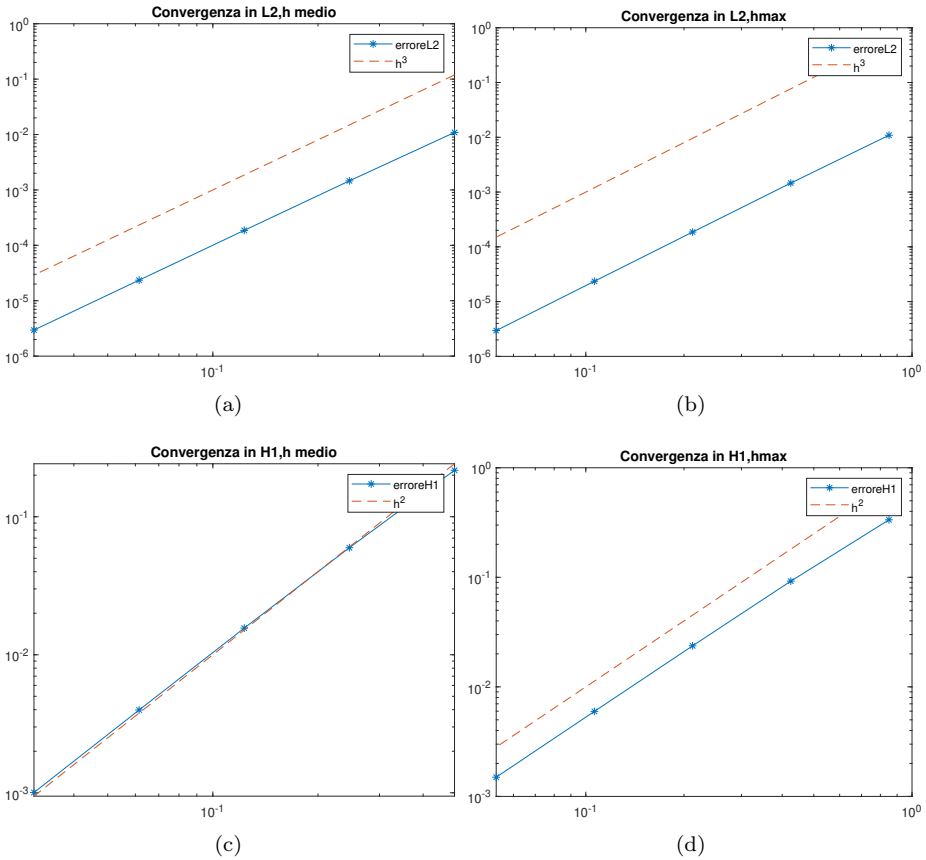
L'errore totale commesso usando il metodo degli elementi finiti è una somma di più errori con peso diverso.

Infatti:

$\|u - u_h\|_{1,\Omega} \simeq$ (errore di approssimazione per l'uso di $V_h \subset V$) + (errore dovuto alla formula di quadratura utilizzata) + (errore dovuto all'aritmetica di floating point).

Esiste inoltre un teorema che afferma che se la formula di quadratura ha grado di precisione $gdp \geq 2k-1$, allora l'errore introdotto usando questa formula è trascurabile rispetto all'errore

gia commesso prendendo $V_h \subset V$. Per verificare che l'ordine di convergenza dell'errore sia $H^1 che L^2 non cambia prendendo una formula di quadratura con grado di precisione gdp \geq 2k-1$ abbiamo calcolato tali errore con una formula di quadratura $fdq =' degree4'$. Riportiamo di seguito tali grafici che possiamo confrontare con i grafici sopra riportati con una formula di quadratura $fdq = cubici$.



(a) (b) (c) (d)

Figure 2: Convergenza in H1 e L2 con fdq degree 4

Dal confronto tra i grafici con formula di quadratura diverse osserviamo che per sia per l'errore H1 che per l'errore L2 gli ordini di convergenza sono pressochè identici.

1.5 Calcolo convergenza in $L^2(\Omega)$ e in $H^1(\Omega)$ quando la soluzione esatta è un polinomio di grado k

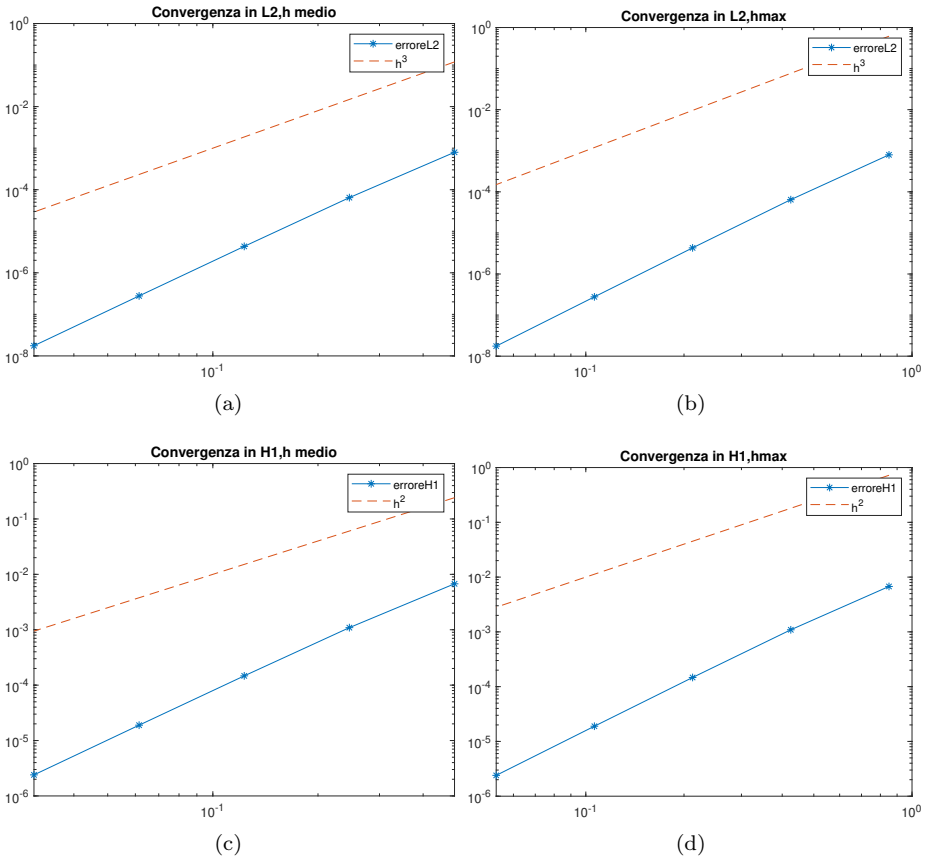
Nel caso in cui la soluzione esatta è presa all'interno dello spazio V_h^k, l'errore dovuto all'approssimazione dello spazio infinito dimensionale V con la spazio finito dimensionale V_h^k viene eliminato completamente, mostrando quindi l'errore commesso nell'approssimare l'integrale con una formula di quadratura.

Sempre considerando il caso $K = 2$, per verificare questo punto abbiamo preso una *uesatta* polinomiale data da:

$$u = x^2 + y^2 + xy + 1$$

Tramite lo script *calcolo_effe* abbiamo calcolato la giusta f e sistemato le condizioni al bordo. Vengono riportati di seguito i grafici degli errori utilizzando una formula di quadratura $fdq = cubici$.

Possiamo osservare che effetivamente che l'ordine di convergenza è diminuito drasticamente. Infatti l'errore che si osserva è quello dovuto solo all'approssimazione della formula di quadratura(più quello dovuto all'errore floating point che pero è trascurabile).Questo è coerente con la teoria.



(a) (b)

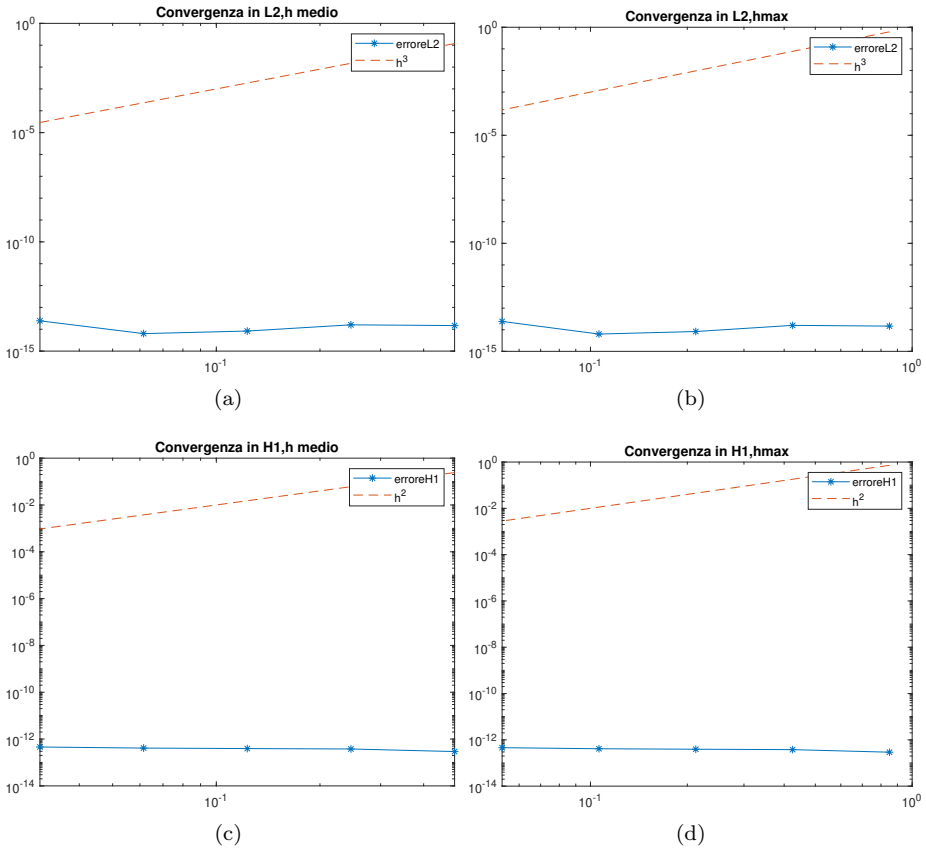
(c) (d)

Figure 3: Convergenza in H1 e L2 con fdq cubici

Come ultimo punto abbiamo calcolato esattamente l'integrale utilizzando una formula di quadratura uguale al grado del polinomio della funzione integranda relativo all'elemento di matrice A. In questo modo è possibile osservare l'errore floating point altrimenti impossibile da osservare perchè trascurabile rispetto agli altri due errori. Avendo fissato il coefficiente K come polinomio di grado 2 ed essendo nel caso dei fem di grado 2, serve una formula di quadratura di grado di precisione 4. Abbiamo fatto girare i codici impostando la formula di quadratura $fdq = degree4$ ottenendo i risultati riportati sotto(dopo le conclusioni).

1.6 Conclusioni

Dai grafici sopra riportati abbiamo potuto osservare come l'ordine di convergenza vada come h^3 per l'errore L2 e come h^2 per l'errore H1. Inoltre utilizzando una soluzione esatta polinomiale si puo notare che l'ordine di grandezza dell'errore cala vistosamente, segno che effettivamente l'errore dovuto all'approssimazione dello spazio V con lo spazio di approssimazione V_h si annulla, facendo emergere l'errore dovuto alla formula di quadratura utilizzata per il calcolo dell'integrale. Infine, utilizzando una formula di quadratura che calcola l'integrale esattamente, abbiamo osservato l'errore floating point che ha un ordine di convergenza di $10^{-12/-15}$.



(a) (b) (c) (d)

Figure 4: Convergenza in H1 e L2 con fdq degree 4

2 Esercizio 2

La risoluzione dell'esercizio si può strutturare nei seguenti punti chiave:

1. Scelta del dominio Ω e calcolo dell'area e del perimetro;
2. Generazione di N autovalori discreti;
3. Funzione di Weil e analoga discreta SommaAutov;

2.1 Scelta del dominio Ω e calcolo dell'area e del perimetro

Per questo esercizio abbiamo considerato tre domini diversi, in ordine crescente di elaborazione. Questi sono:

- il quadrato standard di lato 1;
- un rettangolo con un buco;
- un dominio più fantasioso, che raffigura la testa di un cane, con 4 buchi.

Gli ultimi due domini sono rappresentati in figura 1 e 2. Per entrambi i domini il calcolo dell'area e del perimetro è stato fatto tramite una funzione *Calcolo_area_per* a cui abbiamo passato in input:

- le coordinate dei nodi, ossia xnodo e ynodo;
- la matrice dei nodi degli elementi;
- la matrice dei nodi dei lati

Il codice relativo al calcolo dell'area è stato preso dalla funzione FemPk. A questo abbiamo aggiunto l'implementazione relativa al calcolo del perimetro la cui logica è la seguente:

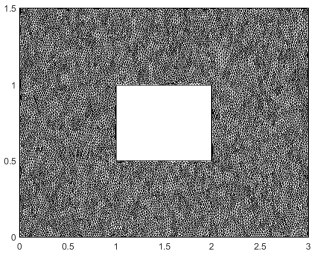


Figure 5: dominio rettangolo con buco

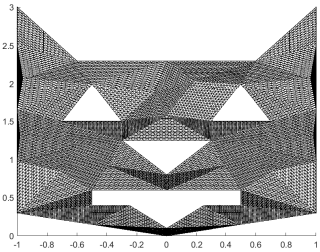


Figure 6: dominio cagnolino

Tra tutti i lati nella matrice degli edge, consideriamo solo quelli di bordo, ossia quelli per cui si ha $edgemarker = 1$. Questo è stato fatto mediante l'istruzione $find$ a cui abbiamo passato la giusta condizione sull'$edgemarker$. Per ogni edge con questa proprietà, andiamo a richiamare le coordinate dei nodi con indice relativo. A questo punto abbiamo determinato la lunghezza del lato sfruttando la funzione *sqrt*.

2.2 Generazione di N autovalori discreti

Questo è l'unico punto in cui bisogna distinguere tra il caso di autovalori calcolati con i Fem di grado $k = 1$ e quelli dovuti a Fem di grado $k = 2$. La differenza sostanziale sta nel raffinamento della mesh.

2.2.1 Raffinamento della mesh

Nel primo caso, al primo ciclo di successione si crea la struttura dei P1 con la funzione *readmesh* e *readedge*, e dal secondo ciclo in poi richiamo la funzione $refine$ che mantiene la struttura di partenza;
Nel caso dei P2 invece il raffinamento della mesh è stato strutturato come segue:
Alla prima successione della mesh abbiamo usato la funzione *readmesh*2 che a sua volta richiama prima le funzioni *readmesh* e la *readedge* per creare la struttura dei P1 e successivamente, dato che il caso $k = 2$ necessita di una struttura P2 per la costruzione della matrice, cioè a 6 nodi per triangolo, richiama la funzione $refine$. Quest'ultima però genera 3 nuovi nodi per triangolo che aggiunge in coda all'ultimo elemento della matrice dei nodi degli elementi, lasciando invariata la struttura di partenza. Quindi il passo successivo è stato creare una matrice a 6 colonne tante quanti sono i nodi per elemento necessari alla struttura P2, e aggiungere alle nuove colonne i nodi aggiunti inizialmente in coda alla vecchia matrice.
Dal secondo ciclo in poi abbiamo richiamato 2 volte la funzione $refine$: la prima necessaria per il vero e proprio raffinamento della mesh, e la seconda per ricreare la struttura dei P2 non salvata al ciclo precedente.

2.2.2 Assemblaggio matrici

Per la costruzione delle matrici Ae e Mh relative rispettivamente ai due integrali

$$\int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i dx \qquad \int_\Omega \varphi_j\varphi_i dx$$

ci siamo serviti delle funzioni già esistenti per l'assemblaggio della matrice relativo al sistema lineare $Ax = b$ ossia:

- funzione $FemPk$ per la costruzione delle matrici sfruttando la tecnica dell'assemblaggio;
- $nodi\_dirichlet$;
- $sistema\_effettivo$ per estrarre dalla matrice NxN, comprensiva di nodi di bordo, la matrice dei soli nodi interni;

Per tutte e 3 le funzioni abbiamo aggiunto il codice relativo alla costruzione della matrice Mh.
Per il caso $k = 1$ l'elemento di matrice

$$Mh(i,j) = \int_T \varphi_j\varphi_i dx$$

è stato calcolato esattamente ottenendo i seguenti risultati:
Sia $|T|$ l'area del triangolo considerato, per l'elemento di matrice diagonale si ottiene

$$Mh(i,i) = \frac{|T|}{6}$$

In tutti gli altri casi, in cui $i \neq j$,

$$Mh(i,j) = \frac{|T|}{12}$$

Per il caso $k = 2$ l'integrale dell'elemento di matrice è stato calcolato utilizzando una formula di quadratura di grado di precisione $gdp \geq 2k - 1$. In particolare abbiamo usato le seguenti formule di quadratura:

- 'punti medi';
- 'cubici';
- 'degree 4';

2.2.3 Calcolo autovalori discreti

Una volta ottenute le due matrici Ae e Mh abbiamo calcolato N autovalori discreti mediante la funzione $eigs$ a cui abbiamo passato i seguenti paramentri:

$$Ae, Mh, N, smallest$$

dove con la proprietà $sigma = smallest$ si chiede alla funzione di restituire gli N autovalori piu piccoli in ordine crescente. Abbiamo creato una matrice aut_disc con numero di colonne uguale al numero di cicli di successione della mesh e con numero di righe uguale a N.
Con la proprietà $sigma = smallest$ abbiamo però riscontrato dei problemi sia al variare dei cicli di successione diversi e sia modificando la formula di quadratura nel caso $k = 2$.
In particolare, dal terzo ciclo di successione in poi, gran parte degli autovalori avevano valore NaN.
Abbiamo provato quindi a modificare la proprietà $sigma$ con $sigma = SM$ e $sigma = smallestabs$.
Questa modifica ha portato alla risoluzione del problema per qualsiasi successione eccetto per la prima: al primo ciclo di successione infatti, da un certo autovalore fissato in poi abbiamo ottenuto valori inf. Dato che questi autovalori partivano dalla posizione 78, e quindi si ottenevano comunque un discreto numero di autovalori corretti, abbiamo deciso di lasciare questa miglioria.
Inoltre questa modifica ha risolto anche il problema sulle formule di quadratura: anche in questo caso si otteneva infatti una matrice degli autovalori con intere colonne con valore NaN. Ottenuta la matrice aut_disc degli autovalori, abbiamo controllato che gli autovalori fossero $'giusti'$ e questo è stato possibile confrontando tra loro gli autovalori dell'N-esima riga. Per i motivi spiegati sopra abbiamo dovuto restringere il numero di autovalori a 78. Nonostante questa restrizione, possiamo

osservato come gli autovalori finali della prima colonna si discostassero parecchio dagli altri autovalori della medesima riga. Dalla seconda colonna gli autovalori si avvicinano molto e quindi possiamo considerarli 'giusti'.

2.3 Funzione di Weil e analogo discreto SommaAutov

Ottenuti gli autovalori, abbiamo calcolato la formula di Weil

$$N(s) = \frac{|\Omega|}{4\pi}s - \frac{|\partial\Omega|}{4\pi}\sqrt{s}$$

Questo è stato fatto creando una funzione a cui si passa l'area $|\Omega|$ e il perimetro $|\partial\Omega|$ calcolate all'inizio del programma dalla funzione $Calcolo\_area\_per$.
Più articolato è l'algoritmo della funzione SommaAutov che prende in input:

- la matrice degli autovalori aut_disc;
- il numero del ciclo di successione corrente $isucc$;
- l'array s che rappresenta la variabile indipendente.
- un indice j che, all'interno di un ciclo for, serve per considerare ad ogni ciclo un elemento di s.

Il fine della funzione è contare il numero di autovalori il cui valore è minore di $s(j)$. Per fare ciò si inizializza un contatore che,per ogni valore $s(j)$, si incrementa di una unità se l'autovalore è minore di s, altrimenti usciamo dal for usando l'istruzione $break$.

2.4 Risultati e considerazioni finali

Come esempio riportiamo di seguito i grafici per $k = 2$ relativi ad un raffinamento annidato pari a 5 usando come dominio il rettangolo con buco.
Possiamo osservare che nel primo grafico, relativo alla mesh creata da triangle, le due funzioni si discostano molto. Man mano che si raffina la mesh le due funzioni di avvicinano fino a sovrapporsi.

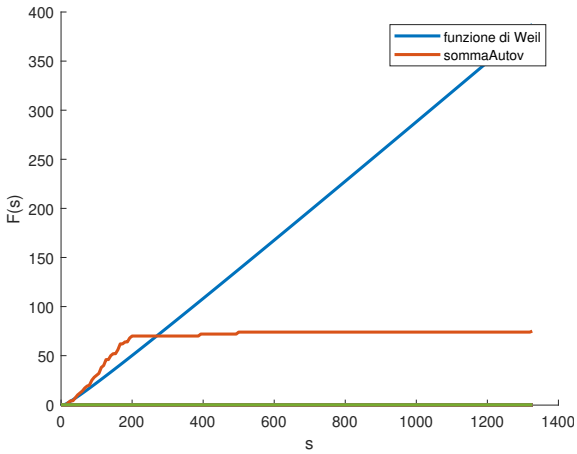


Figure 7: grafico mesh_nested 1

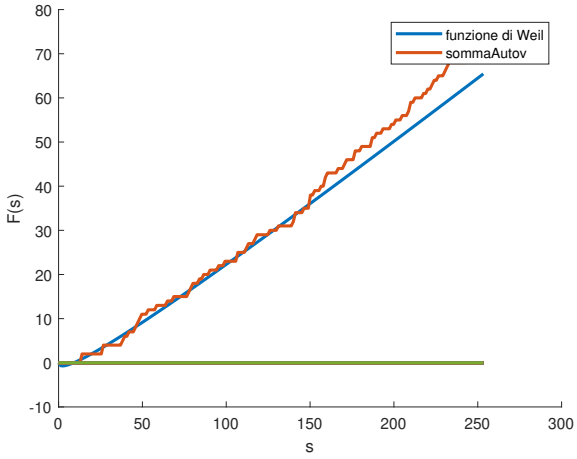


Figure 8: grafico mesh_nested 2

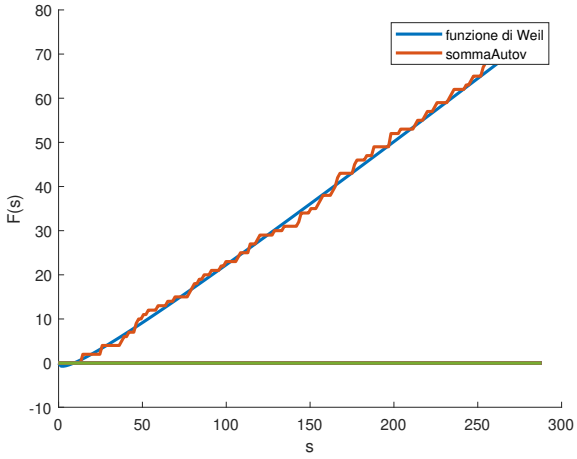


Figure 9: grafico mesh_nested 3

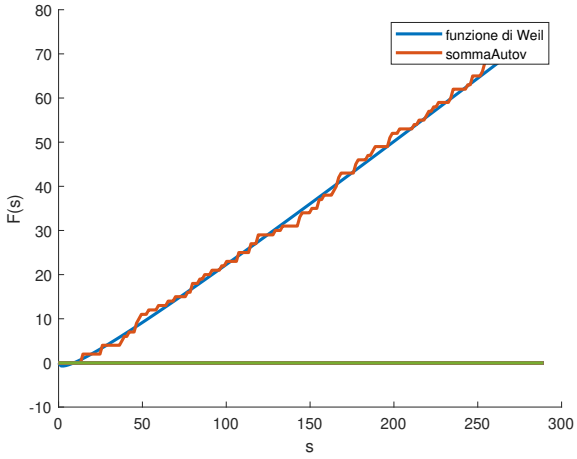


Figure 10: grafico mesh_nested 3

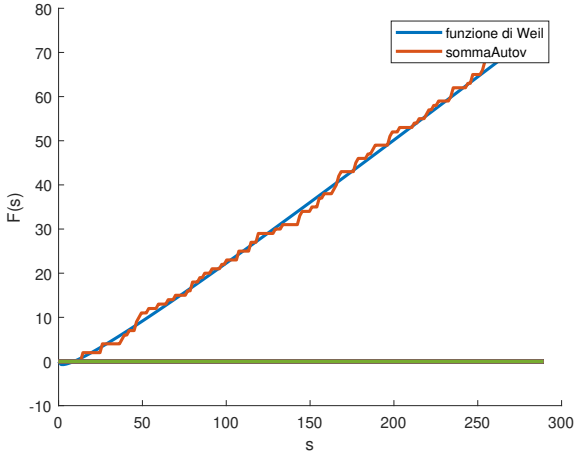


Figure 11: grafico mesh_nested 3