Relazione Approssimazione Di Equazioni Differenziali

Pirola Zaltieri Stella Saporito Francesco

November 16, 2016

Contents

Laurea Magistrale in Matematica **Anno Accademico 2015-2016**

Relazione per il corso di Approssimazione di Equazioni Differenziali

Prof. Alessandro Russo

La relazione ha lo scopo di studiare l'ordine di convergenza in L^2 e in H^1 del metodo degli elementi finiti in due dimensioni basato sui triangoli e sui polinomi di grado 2.

Il punto di partenza sono i codici e le note reperibili **qui**.

✎ Scrivere un codice Matlab che risolva l'equazione differenziale alle derivate parziali

$$-\operatorname{div}(c\nabla u) + \boldsymbol{\beta} \cdot \nabla u + \alpha u = f \quad \text{in } \Omega$$

con condizione al bordo di Dirichlet $u = g$ su $\Gamma = \partial\Omega$ utilizzando elementi finiti di grado 2.

Parametrizzare il codice prevedendo di usare varie formule di quadratura per il calcolo della matrice di stiffness e del termine noto.

Come dimostrato a lezione, se si calcolano gli integrali esattamente, in questo caso l'errore nella seminorma $H^1(\Omega)$ è $O(h^2)$ mentre in norma $L^2(\Omega)$ è $O(h^3)$, dove h è il massimo dei diametri dei triangoli.

Nel caso di elementi finiti polinomiali di ordine k, si dimostra che l'errore commesso nell'approssimazione dei coefficienti della matrice e del termine noto è trascurabile se il grado di precisione delle formule di quadratura utilizzate è (almeno) $2k - 1$, quindi $2 \times 2 - 1 = 3$ in questo caso.

✎ Verificare che se si usa una qualunque formula di quadratura di grado di precisione 3 si ottiene l'ordine previsto in L^2 e in H^1.

✎ Verificare inoltre che se si utilizza una formula di quadratura di grado di precisione maggiore di 3 gli ordini di convergenza non cambiano.

Per queste verifiche, procurarsi come spiegato a lezione una problema di cui si conosca *a priori* la soluzione esatta (regolare) u_{esatta}. Scegliere quindi u_{esatta} e scegliere poi delle funzioni c, $\boldsymbol{\beta}$, α, regolari con $c(x, y) \geq c_0 > 0$ e $-\frac{1}{2}\operatorname{div}\boldsymbol{\beta} + \alpha \geq 0$ in modo da assicurare le ipotesi di coercività e regolarità del problema. Definire f usando l'equazione e porre $g = u_{esatta}$.

Non limitarsi ad usare soluzioni esatte polinomiali perché potrebbero avere degli ordini di convergenza diversi (in generale più alti) di quelli previsti dalla teoria.

Usare come h sia la media di tutti gli edge, sia il massimo di tutti gli edge della triangolazione (che coincide col massimo dei diametri di tutti i triangoli). Le curve dell'errore nel caso in cui h è la media sono di solito più lineari.

Usate sia mesh generate da triangle sia delle mesh regolari (sul sito del corso c'è scritto come procurarsele e come usarle). Nel caso delle mesh regolari l'andamento delle curve di convergenza dovrebbe essere più lineare.

Per calcolare la norma L^2 dell'errore, si proceda in questo modo:

$$\|u_{esatta} - u_h\|^2_{L^2(\Omega)} = \int_\Omega |u_{esatta} - u_h|^2 = \sum_T \int_T |u_{esatta} - u_h|^2.$$

Ci siamo quindi riportati a integrali sui triangoli che possono essere calcolati analogamente alla matrice di stiffness ancora tramite formule di quadratura sul triangolo di riferimento. In questo caso occorre utilizzare una formula di quadratura che abbia grado di precisione pari almeno all'ordine di convergenza atteso più uno (se no l'errore nel calcolo dell'integrale si mangia l'errore di approssimazione!).

Per l'errore in H^1 si procede in modo analogo.

Riportare su un grafico in scala logaritimica h per varie mesh e l'errore corrispondente e verificare che le pendenze (cioè gli ordini) siano quelli previsti caso per caso.

Chapter 2

Elementi Finiti Per Problemi Ellittici

2.1 Introduzione

In questo capitolo richiamiamo i concetti base del metodo agli elementi finiti per risolvere problemi ellittici del secondo ordine.
Il problema differenziale generale che affronteremo è del tipo:

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

dove $\Omega \in \Re^n$ e dove con $\partial\Omega$ intendiamo il bordo relativo al dominio Ω. I vari termini dell'equazione possono essere caratterizzati nel modo seguente:

- **Termine Di Diffusione** $-div(c\nabla u)$
- **Termine Di Avezione** $\beta \cdot \nabla u$
- **Termine Di Reazione** αu

Al bordo abbiamo considerato la condizione non omogenea di Dirichlet $u = g$. Sono possibili altri tipi di condizioni al bordo, ad esempio:

- **Neumann** Condizione sulla derivata normale al bordo della funzione.
- **Cauchy** Impone al bordo sia una condizione di Dirichlet che una di Neumann.
- **Robin** Impone al bordo una condizione ottenuta come combinazione lineare di una condizione di Dirichlet e di una di Neumann.

L'equazione ellittica del secondo ordine è molto importante per la modellizzazione di problemi stazionari (ovveero indipendenti dal tempo), e infatti molte altre equazioni si riducono ad una riconducibile al caso ellittico nel regime stazionario [1]. Forniamo alcuni esempi fisici modellizzati da questa equazione:

Example 1 (Equazione di Poisson)**.**
Il caso più semplice è l'equazione di Poisson:

$$\begin{cases} -\Delta u = f & \Omega \\ u = g & \partial\Omega \end{cases}$$

dove Δ è l'operatore di Laplace, definito come:

$$\Delta = div\nabla$$

[1] Ad esempio le equazioni di tipo parabolico e iperbolico sono equazioni ellittiche nelle variabili spaziali con l'aggiunta di termini temporali

Questa equazione modellizza ad esempio problema del calore stazionario [Sal10, Capitolo 3] o il problema generale dell'elettrostatica in presenza di cariche [Jac98, Capitoli 1,2,3]. ∎

Example 2 (Equazione di Fick Stazionaria)**.**
L'equazione di Fick descrive la variazione di concentrazione nei materiali in cui è presente diffusività molecolare ma non termica. Nel caso stazionario (ovvero che studia solo la diffusione spaziale senza considerare il tempo), abbiamo:

$$\begin{cases} -D\Delta u + v\nabla u = f & \Omega \\ \partial_\eta u = 0 & \partial\Omega \end{cases}$$

dove al bordo sono state supposte condizioni di Neumann omogenee, a significare che non c'è diffusione attraverso il bordo del dominio (caso che invece avviene se ad esempio il bordo è poroso). ∎

Gli esempi qui posti sono tutti riguardanti quantità incognite scalari, che è il caso che considereremo in seguito. Il ragionamento può essere esteso a sistemi di equazioni ellittiche (ovvero a grandezze vettoriali). Ad esempio [BS07, Capitolo 11] mostra un'applicazione delle tecniche qui presentate al problema bidimensionale dell'elasticità planare.

2.2 Metodi Variazionali

Vogliamo dunque risolvere un problema del tipo 2.1. Esso in particolare per essere ben definito richiedere ad esempio che la soluzione u sia almeno di classe $C^2(\Omega)$. Questo limita molto la capacità di modellizzazione per problemi fisici reali (ad esempio che presentano discontinuità). Inolte non è detto che sia possibile trovare una soluzione in forma chiusa del problema. Per provare ad ovviare a questi problemi proviamo a passare ad una formulazione equivalente detta formulazione variazionale o debole, che permette alla funzione u di essere meno regolare di quanto previsto dal problema differenziale. I riferimenti principali per questa sezione sono [Bre10] e [Eva90].
Consideriamo inizialmente il seguente problema omogeneo:

$$\begin{cases} -div(c\nabla u) + \beta \cdot \nabla u + \alpha u = f & \Omega \\ u = 0 & \partial\Omega \end{cases}$$

ricordiamo inanzitutto la definizione di derivata debole:

Definition 2.2.1 (Derivata Debole).
Per la formula di integrazione per parti abbiamo che se $f \in C^{|\alpha|}(\Omega)$ e $g \in C_0^{|\alpha|}(\Omega)$, con α multi-indice, allora vale:

$$\int_\Omega (\partial^\alpha f)\, g \, dx \quad = \quad (-1)^{|\alpha|} \int_\Omega f\, (\partial^\alpha g)\, dx$$

da cui, se f è localmente integrabile su Ω, allora diciamo che una funzione g è la sua α-derivata debole, definita (quasi ovunque) come

$$g = D^\alpha f$$

se $\forall \phi \in C_0^{|\alpha|}(\Omega)$ abbiamo

$$\int_\Omega g\, \phi \, dx \quad = \quad (-1)^{|\alpha|} \int_\Omega f\, (\partial^\alpha \phi)\, dx$$

∎

il concetto di derivata debole risulta quindi un'estensione a quello di derivazione classica, infatti ogni funzione di classe C^k ha k-derivata debole. Attraverso la definizione di derivata debole possiamo definire lo spazio di Sobolev $\mathbb{H}^1(\Omega)$:

Definition 2.2.2 ($\mathbb{H}^1(\Omega)$)**.**

$$\mathbb{H}^1(\Omega) \quad = \quad \left\{ v \in \mathbb{L}^2(\Omega) : \quad \nabla v \in \mathbb{L}^2(\Omega) \right\}$$

dove il gradiente è inteso come derivazione debole analogamente a come definito sopra. Questo spazio risulta essere uno spazio di Hilbert con prodotto scalare:

$$< v, v' >_{\mathbb{H}^1(\Omega)} \quad = \quad < v, v' >_{\mathbb{L}^2(\Omega)} + < \nabla v, \nabla v' >_{\mathbb{L}^1(\Omega)}$$

ovvero con norma indotta dal prodotto scalare:

$$|| \, v \, ||_{\mathbb{H}^1(\Omega)} \quad = \quad || \, v \, ||_{\mathbb{L}^2(\Omega)} + || \, \nabla v \, ||_{\mathbb{L}^2(\Omega)}$$

■

Questo spazio è dunque l'insieme delle funzioni di $L^2(\Omega)$ con derivate deboli prime (espresse tramite il gradiente) anch'esse in $L^2(\Omega)$. In questo spazio (o in un suo sottospazio) andremo a cercare le soluzioni deboli della formulazione variazionale.
L'idea dietro alla formulazione debole è che il problema differenziale è scritto localmente. Vogliamo però scriverlo in forma globale, in modo da poter considerare funzioni meno regolari, attraverso il concetto di derivata debole.

Theorem 2.2.1 (Problema Ellitico In Forma Variazionale)**.**
Il problema ellittico 2.2 può essere riscritto in forma variazionale nel seguente modo: Trovare $u \in \mathbb{V} = \mathbb{H}^1(\Omega)$ tale che:

$$\int_\Omega c \, \nabla u \nabla v \; dx \; + \; \int_\Omega (\beta \cdot \nabla u) \, v \; dx \; + \; \int_\Omega \alpha \, u \, v \; dx \quad = \quad \int_\Omega f \, v \; dx \qquad \forall v \in \mathbb{V} = \mathbb{H}^1(\Omega)$$

dove u è detta soluzione debole del problema variazionale.

Dimostrazione
Condideriamo l'equazione

$$-div(c\nabla u) \; + \; \beta \cdot \nabla u \; + \; \alpha u \; = \; f$$

e moltiplichiamo entrambi i membri per una funzione $v \in C^\infty(\Omega)$:

$$-div(c\nabla u) \, v \; + \; (\beta \cdot \nabla u) \, v \; + \; \alpha u v \; = \; f \, v$$

integriamo dunque su tutto il dominio Ω rispetto alla misura di Lebesgue dx del dominio:

$$- \int_\Omega div(c\nabla u) \, v \; dx \; + \; \int_\Omega (\beta \cdot \nabla u) \, v \; dx \; + \; \int_\Omega \alpha \, u \, v \; dx \; = \; \int_\Omega f \, v \; dx$$

consideriamo il primo integrale. Per le proprietà della divergenza se f è una funzione scalare e se $\mathbb{F}$ è un campo vettoriale, allora vale:

$$div(f\mathbb{F}) \; = \; \nabla f \cdot \mathbb{F} \; + \; f div(\mathbb{F})$$

considerando dunque $f = v$ e $\mathbb{F} = c\nabla u$, abbiamo

$$div(vc\nabla u) = \nabla v(c\nabla u) + v\, div(c\nabla u)$$

ovvero

$$v\, div(c\nabla u) = div(v\, c\, \nabla u) - c\, \nabla v \nabla u$$

sostituendo nell'integrale abbiamo

$$-\int_\Omega div(c\nabla u)\, v\; dx = \int_\Omega c\, \nabla v \nabla u\; dx - \int_\Omega div(v\, c\nabla u)\; dx$$

possiamo quindi applicare il teorema della divergenza

$$\int_\Omega div(v\, c\nabla u)\; dx = \int_{\partial\Omega} (v\, c\nabla u) \cdot \eta \;\; d\sigma(x)$$

dove η è il versore normale e $d\sigma(x)$ è la misura superficiale dell'ipersuperficie $(n-1)$-dimensionale $\partial\Omega$ [2]. La funzione u è però nulla sul bordo, e quindi l'integrale su $\partial\Omega$ è nullo. Otteniamo cosi la forma debole:

$$\int_\Omega c\, \nabla u \nabla v\; dx + \int_\Omega (\beta \cdot \nabla u)\, v\; dx + \int_\Omega \alpha\, u\, v\; dx \quad = \quad \int_\Omega f\, v\; dx$$

dobbiamo però caratterizzare la regolarità delle funzioni u, v e f presenti, affinche questa forma debole sia ben definita. Inanzitutto sia u che v devono avere derivata debole di ordine 1. Affinchè poi gli integrali siano definiti e convergenti, dobbiamo avere che u, v, ed f devono essere almeno in $L^2(\Omega)$, cosi come i gradienti (intesi nel senso della derivazione debole) di u e v. Ciò implica quindi che sia u che v devono appartenere a $\mathbb{V} = \mathbb{H}^{\mathbb{1}}(\Omega)$. Inoltre abbiamo imposto le condizioni omogeneee di Dirichlet. Possiamo quindi considerare il sottospazio di funzioni di $\mathbb{V} = \mathbb{V}_0 = \mathbb{H}^1_0(\Omega)$:

$$\mathbb{V}_0 = \mathbb{H}^1_0(\Omega) = \left\{ v \in \mathbb{H}^1(\Omega) \, : \, v|_{\partial\Omega} = 0 \right\} \subset \mathbb{H}^1(\Omega)$$

da cui risulta che

- $f \in L^2(\Omega)$
- $u,\, v \in \mathbb{H}^1_0(\Omega) \subset \mathbb{H}^1(\Omega)$

$\blacksquare$

allo stesso modo si può procedere per il caso non omogeneo 2.1, dove però al bordo l'integrale non è più nullo, ma va considerato con $u = g$ a livello di tracce.
Il problema in forma debole 2.2.1, può essere espresso in forma astratta:

Definition 2.2.3 (Problema Astratto)**.**
Trovare $u \in \mathbb{V}$, con $\mathbb{V}$ spazio di Hilbert, tale che

$$a(u, v) = F(v) \qquad\qquad \forall v \in \mathbb{V}$$

dove

- $a : V \times V \to \Re$ è una forma bilineare
- $F : V \to \Re$ è un funzionale lineare, ovvero $F \in \mathbb{V}'$.

[2]La misura superficiale differisce dalla misura di Lebesgue in quanto tiene conto anche della curvatura dello spazio

■

Nel nostro caso particolare 2.2.1 il problema astratto è formato da:

Definition 2.2.4 (Problema ellittico astratto)**.**

- $\mathbb{V} = \mathbb{H}^1(\Omega)$
- $a(u, v) \;=\; \int_\Omega c\,\nabla u \nabla v\; dx \;+\; \int_\Omega (\beta \cdot \nabla u)\, v\; dx \;+\; \int_\Omega \alpha\, u\, v\; dx$
- $F(v) \;=\; \int_\Omega f\, v\; dx$

■

Il lemma di Lax-Milgram fornisce un risultato di esistenza e unicità della soluzione (debole) al problema astratto, che coincide dunque con l'esistenza e unicità della soluzione di 2.2.1 e di 2.1:

Theorem 2.2.2 (Lax-Milgram - 1954)**.**
Esiste ed è unica la soluzione debole $u \in \mathbb{V}$ di Hilbert per il problema astratto 2.2.3 se:

- *$a(u, v)$ è continua:* $\quad a(u, v) \;\leq\; M \,||\, u \,||_{\mathbb{V}} \,||\, v \,||_{\mathbb{V}}$
- *$a(u, v)$ è coerciva:* $\quad a(v, v) \;\geq\; c \,||\, v \,||^2_{\mathbb{V}}$
- *$F(v)$ è continua:* $\quad F(v) \;\leq\; C \,||\, v \,||_{\mathbb{V}}$

Questo lemma è stato in seguito generalizzato da Babuska [Bab71], in modo da considerare due spazi di appartenenza diversi per u e per v e indebolendo la richiesta sulla coercività (richiedendo la coercività debole).
Nel caso in cui $a(u, v)$ sia anche simmetrica, tale forma è un prodotto scalare, e il lemma di Lax-Milgram deriva direttamente dal teorema di rappresentazione di Riesz.
Nel caso particolare di 2.2.4, si può dimostrare che affinchè valgano le condizioni delle ipotesi del lemma di Lax-Milgram 2.2.2, le funzioni caratterizzanti il problema devono rispettare le seguenti:

- $c(\underline{x}) \in L^\infty \qquad c_{max} \;\geq\; c(\underline{x}) \;\geq\; c_{min} \;>\; 0$
- $\alpha(\underline{x}) \in L^2(\Omega),\; \beta(\underline{x}) \in C^1(\Omega) \qquad \alpha \;-\; div\beta \;=\; 0$
- $f(\underline{x}) \in L^2(\Omega)$

MIGLIORARE DESCRIZIONE CASO NON OMOGENE0
FINE SEZIONE

2.3 Metodo Galerkin

Abbiamo visto che il problema astratto 2.2.4 ammette soluzione unica se la forma bilineare $a(u, v)$ è continua e coerciva, e se la forma lineare $F(v)$ è continua per il lemma di Lax-Milgram 2.2.2. Non esiste però un metodo generale per trovare la soluzione esatta al problema astratto in forma chiusa su un qualunque dominio Ω (soluzione che non è detto che sia formulabile in forma chiusa). Per risolvere il problema dobbiamo quindi tentare di stimare la soluzione esatta con una soluzione approssimata, valutandone poi l'errore di approssimazione. La principale bibliografia per questa sezione è formata da [BS07] e

da [Cia78] e [Qua08].
Uno dei metodi principali che consente questo processo è quello di Galerkin [3]:

Definition 2.3.1 (Metodo Galerkin)**.**
Il metoodo Galerkin consiste nell'approssimare lo spazio continuo $\mathbb{V}$ con una famiglia di spazi $\mathbb{V}_h$, dipendenti da un parametro positivo h, tali che:

- $\mathbb{V}_h \subset \mathbb{V}$
- $dim\,(\mathbb{V}_h) \;=\; N_h \;<\; \infty \qquad \forall h > 0$ (Ovvero ogni $\mathbb{V}_h$ è uno spazio discreto)

In questo modo possiamo riscrivere il problema astratto in forma discreta sul singolo spazio $\mathbb{V}_h$:

$$Trovare\ u_h \in \mathbb{V}_h,\ tale\ che\ a(u_h, v_h) = F(v_h) \qquad \forall v_h\ \in \mathbb{V}_h$$

detta equazione di Galerkin. [4] ■

dato che $\mathbb{V}_h \subset \mathbb{V}$, continuano a valere le ipotesi del lemma di Lax-Milgram 2.2.2, e quindi la soluzione al problema discreto esiste ed è unica. Inoltre vale la seguente relazione tra la soluzione del problema continuo e quella del problema approssimato:

Definition 2.3.2 (Ortogonalità Di Galerkin)**.**
Sia u la soluzione esatta del problema continuo e sia u_h la soluzione del problema discreto. Definito l'errore ϵ_h come

$$\epsilon_h \;=\; u \;-\; u_h$$

allora abbiamo che, sottraendo i due problemi membro a membro

$$a(u, v) \;-\; a(u_h, v_h) \;=\; f(v) \;-\; f(v_h)$$

possiamo perè restringere le funzioni v allo spazio $\mathbb{V}_h$, ovvero considerare

$$a(u, v_h) \;-\; a(u_h, v_h) = f(v_h) \;-\; f(v_h) = 0)$$

per linearità della forma a abbiamo

$$a(u, v_h) \;-\; a(u_h, v_h) \;=\; a(u \;-\; u_h, v_h) \;=\; a(\epsilon_h, v_h) = 0$$

nel caso in cui la forma $a(\cdot, \cdot)$ sia simmetrica (ovvero è un prodotto scalare), abbiamo che l'errore ϵ_h è ortogonale ad ogni funzione $\forall v_h \in \mathbb{V}_h$, e quindi è ortogonale all'intero spazio $\mathbb{V}_h$. ■

E' possibile inoltre dare una stima dell'errore compiuto, attraverso il lemma di Cèa:

Theorem 2.3.1 (Lemma di Cèa)**.**
L'errore compiuto con l'approssimazione tramite il problema discreto permette la seguente stima:

$$||\;\; u \;-\; u_h \;\;||_{\mathbb{V}(\Omega)} \quad\leq\quad \frac{M}{c}\; Inf_{v_h \in \mathbb{V}_h}\, ||\;\; u \;-\; v_h \;\;||_{\mathbb{V}(\Omega)}$$

dove M è la costante della continuità e c quella della coercività di $a(\cdot, \cdot)$. ■

[3]Un altro metodo rilevante, ma più semplice è quello delle differenze finite, in cui si approssimano direttamente le derivate sui nodi per riformulare l'equazione in modo approssimato

[4]Notiamo che la forma dell'equazione non cambia tra il problema continuo e quello discreto, cambiano invece gli spazi con cui lavoriamo

L'approssimazione tramite lo spazio $\mathbb{V}_h$ consente inoltre di trasformare il problema continuo in un sistema lineare di equazioni. Dato che lo spazio $\mathbb{V}_h$ è un sottospazio vettoriale finito, possiamo considerarne una base

$$\overline{\phi} = \{\phi_i\} \qquad\qquad i = 1, ..., n = \dim V_h$$

e quindi possiamo sviluppare la funzione u_h rispetto ad essa:

$$u_h = \sum_{i=j}^{n} u_j \ \phi_j$$

è inoltre possibile dimostrare che l'approssimazione vale anche considerando il problema solo riferito alle basi[5], ovvero sostituendo alle funzioni v_h le funzioni di base ϕ_i

$$a \ \left(\sum_{j=1}^{n} u_j \ \phi_j \ , \ \phi_i \right) \ = \ F \ (\phi_i)$$

Sfruttando la linearità di $a(\cdot, \cdot)$ otteniamo:

$$\sum_{j=1}^{n} u_j \ a \ (\phi_j \ , \ \phi_i) \ = \ F \ (\phi_i)$$

e sommando su tutte le funzioni di base (ovvero rispetto all'indice i):

$$\sum_{i=1}^{n} \sum_{j=1}^{n} u_j \ a \ (\phi_j \ , \ \phi_i) \ = \ \sum_{i=1} F \ (\phi_i)$$

ovvero abbiamo ridotto il problema ad un sistema lineare di equazioni, definendo dunque il problema astratto discreto in forma matriciale:

Definition 2.3.3 (Problema Astratto Discreto In Forma Matriciale)**.**
Trovare $\underline{u} \in \Re^n$ tale che

$$K_h \, \underline{u} \ = \ \underline{f}$$

dove abbiamo definito la matrice di stiffness Kh e il vettore termine noto $\underline{f}$ come:

$$(K_h)_{i,j} \ = \ a \ (\phi_j \ , \ \phi_i) \qquad\qquad (\underline{f})_i \ = \ F \ (\phi_i)$$

■

Nel caso particolare del problema ellittico astratto 2.2.4 abbiamo la matrice di stiffness

$$(K_h)_{i,j} \ = \ a(u,v) \ = \ \int_\Omega c \, \nabla\phi_j \nabla\phi_i \ dx \ + \ \int_\Omega (\beta \cdot \nabla\phi_j) \, \phi_i \ dx \ + \ \int_\Omega \alpha \, \phi_j \, \phi_i \ dx$$

e il vettore termine noto:

$$(\underline{f})_i \ = \ F \ (\phi_i) \ = \ \int_\Omega f \, \phi_i \ dx$$

CONTROLLARE LA CORRETTEZZA DEGLI INDICI RISPETTO A QUANTO FATTO A LEZIONE
FINE SEZIONE

[5] Questo in quanto possiamo scrivere ogni funzione v_h dello spazio $\mathbb{V}_h$ come combinazione lineare delle basi

2.4 Approssimazione Con Elementi Finiti

Il metodo di Galerkin fornisce l'idea di approssimare il problema continuo con quello discreto, ma non da vincoli su come fare questa approssimazione. Gli elementi finiti sono un metoodo di Galerkin in cui l'approssimazione viene fatta considerando polinomi su elementi di una partizione (poligonale) del dominio. I testi di riferimento sono [BS07] e da [Cia78] e [Qua08]. In particolare lavoriamo in questa sezione considerando per semplicità un dominio $\Omega \in \Re^2$, con l'ovvia estensione al caso in più di due dimensioni. Il fondamento del metodo agli elementi finiti è di costruire lo spazio discreto $\mathbb{V}_h$ tramite funzioni polinomiali da considerare sugli elementi di una mesh rappresentante Ω.
L'idea di costruire una mesh per il dominio Ω si basa sul concetto topologico di complesso simpliciale euclideo [FP13, Chapter 2], che nel caso particolare bidimensionale permette di considerare la scomposizione di un dominio poligonale in un numero finito di triangoli (e quindi fare una triangolazione del dominio) seguendo le due seguenti proprietà:

- Un vertice di un triangolo $\mathbb{T}_i$ può essere un punto di un qualsiasi altro triangolo $\mathbb{T}_j$ solo se è un vertice dell'altro triangolo. Questo impedisce che un triangolo abbia un verice nel lato di un'altro (ovvero impone la conformità dei triangoli).

- Se l'intersezione degli interni di due triangoli non è vuota, allora i due triangoli devono coincidere. Questa proprietà serve ad escludere triangoli uno interno all'altro o incastonati.

Nel seguito del paragrafo ci riferiremo per semplicità a domini poligonali, in particolare al quadrato di lato unitario con un vertice nell'origine e con due lati sugli assi$x$ e y. Nel caso più generale in cui il dominio non sia poligonale, bisognerà trovare delle tecniche di approssimazione per poter comunque generare una sua triangolazione e quindi crearne una mesh.[6]. Definiamo dunque un elemento finito:

Definition 2.4.1 (Elemento Finito)**.**
Definiamo un elemento finito come una terna $(\mathbb{T}\,,\,\mathbb{P}\,,\,\Sigma)$, dove:

- $\mathbb{T}$ è l'insieme dei triangoli della mesh sul dominio Ω, ovvero è il complesso simpliciale associato[7] al dominio Ω.

- $\mathbb{P}$ è lo spazio locale che utilizziamo per l'approssimazione.

- Σ è l'insieme dei gradi di libertà $\quad \Sigma = \{\Sigma_i : \mathbb{P} \to \Re \quad \text{lineare}\}_{i=1}^{dim\mathbb{P}}$

■

Attraverso gli elementi finiti possiamo costruire lo spazio $\mathbb{V}_h$ come il sottoinsieme di $\mathbb{V}$ tale che i suoi elementi siano localmente (ovvero su ogni triangolo) esprimibili come elementi dello spazio $\mathbb{P}$ e soddisfacenti alcune condizioni ulteriori per garantire la regolarità del problema discreto cosi formulato. In particolare considereremo in seguito elementi cosi definiti:

Definition 2.4.2 (Elementi Finiti Polinomiali)**.**
Considereremo come spazio locale di approssimazione $\mathbb{P} = \mathbb{P}_k(T)$, ovvero lo spazio dei polinomi di grado k sull'elemento T, mentre definiamo lo spazio $\mathbb{V}_h$ come:

$$\mathbb{V}_h \;=\; \left\{\, v \in \mathbb{H}^1_g(\Omega) \,:\, v|_{\partial\Omega} = g \,,\, v|_T \in \mathbb{P}_k(T) \;\; \forall T \in \mathbb{T} \,\right\} \;\subset\; \mathbb{H}^1_g(\Omega) \;=\; \mathbb{V}$$

[6]Questo si basa sull'idea di varietà triangolabile, ovvero omeomorfa ad un poliedro [FP13, Chapter 5]. A livello pratico ciò può essere fatto ad esempio con il metodo isoparametrico [Len86] o con l'analisi isogeometrica che si interfaccia alla progettazione CAD [CHB09]

[7]Non è unico, ma allo stesso dominio Ω possiamo associare più complessi simpliciali

Avremo poi in generale un numero N gradi di libertà, definito come

$$N \;=\; dim\,(\mathbb{P}_k(\Omega)) \;=\; \binom{n+k}{k} \;=\; \frac{(n+k)!}{k!\cdot(n!)}$$

dove n è la dimensione dello spazio Ω. Nel nostro caso particolare $n \;=\; 2$, e quindi sui triangoli avremo per i polinomi di grado k un numero di gradi di libertà pari a:

$$N \;=\; dim\,(\mathbb{P}_k(T)) \;=\; \binom{2+k}{k} \;=\; \frac{(2+k)!}{k!\cdot(2!)} \;=\; \frac{(2+k)\cdot(1+k)\cdot k!}{2\cdot k!} \;=\; \frac{(k+2)\cdot(k+1)}{2}$$

■

Per ricondurci alla definizione data del metodo di Galerkin 2.3.1, si considera una famiglia di spazi $\mathbb{V}_h$ analoghi alla definizione appena data e dipendenti dal parametro positivo h che rappresenta il diametro massimo dei triangoli della mesh. In particolare il metodo agli elementi finiti in versione h consiste nel far tendere questo diametro a zero per riottenere il problema continuo. Quello che si fa nella pratica è un'analisi a posteriori dell'errore, nella quale si fissa una certa tolleranza sull'errore compiuto e si usa una versione adattiva del metodo, ovvero si varia automaticamente il parametro h (e quindi si raffina la partizione) per ottenere una soluzione approssimata con la precisione richiesta. Altre varianti sono possibili, quali il metodo p [BSK81], in cui la triangolazione è fissata, ma si usano polinomi di grado p crescente o il metodo hp [BS94] che usa una fusione tra i metodi due metodi h e p.

Definiamo ora una base per questo spazio $\mathbb{V}_h$ come definito in 2.4.2, in modo da poter esplicitare il problema ellittico debole in forma matriciale 2.3:

Definition 2.4.3 (Base di Lagrange)**.**
Definiamo la base di Lagrange ϕ_i per lo spazio $\mathbb{V}_h$ in corrispondenza del nodo libero v_j come alla funzione:

$$\phi_i(v_j) = \begin{cases} 1 & \text{if } \; i \;=\; j \\ 0 & \text{if } \; i \;\neq\; j \end{cases}$$

In particolare notiamo che il supporto della funzione ϕ_i è formata dai triangoli che circondano il vertice v_i. ■

Usando questa base, la risultante matrice di stiffness K_h oltre ad essere definita positiva [8], risulta essere anche una matrice sparsa [9]. Ciò permette da un lato di ridurre le dimensioni in memoria della matrice e dall'altro di usare metodi appositi per la risoluzione di sistemi lineari con matrice associata sparsa, quali il gradiente coniugato o l'uso di precondizionatori [Saa03].
Definiamo dunque il problema astratto discreto agli elemeti finiti come al caso particolare del problema astratto discreto in forma matriciale 2.3.3 dove consideriamo come basi le basi di Lagrange 2.4.3 e gli spazi cosi come definiti nel caso degli elementi finiti polinomiali 2.4.2. In modo analogo definiamo il problema agli elementi finiti che approssima il problema ellittico in forma debole 2.2.1.

[8]Fatto che vale per ogni base
[9]Che invece vale in particolare per questa base ma non in generale

Chapter 3

Calcolo Esplicito

In questo capitolo mostriamo come fare esplicitamente il calcolo della soluzione approssimata tramite gli elementi finiti. Quello che andiamo a risolvere è dunque il problema 2.3.3 rappresentato dal seguente sistema lineare

$$K_h \, \underline{u} \; = \; \underline{f}$$

che scritto esplicitamente rispetto alla definizione del problema ellittico in forma astratta 2.2.4 porta a dover calcolare

$$\int_\Omega c \, \nabla\phi_j \nabla\phi_i \; dx \; + \; \int_\Omega (\beta \cdot \nabla\phi_j) \, \phi_i \; dx \; + \; \int_\Omega \alpha \, \phi_j \, \phi_i \; dx \quad = \quad \int_\Omega f \, \phi_i \; dx$$

dove $\underline{\phi}$ è la base di Lagrange 2.4.3 mentre c, β, α e f sono i dati del problema.
Dato che la base di Lagrange ϕ_i ha supporto compatto formato dai vertici j vicini al vertice i (ovvero appartenenti ai triangoli adiacenti a quello in cui è contenuto i), possiamo limitarci a fare il calcolo su ogni singolo triangolo considerando che solo i termini con vertici i,j appartenente al triangolo corrente risultano non zero, per poi sommare nella matrice K_h tutti i contributi [1]. Scomponiamo quindi la matrice di stiffness K_h 2.3 in una somma matrici elementari K_h^T tali che:

$$(K_h)_{ij} \quad = \quad \sum_{T \in \mathbb{T}} (K_h^T)_{ij} \quad = \quad \sum_{T \in \mathbb{T}} \left(\int_T c \, \nabla\phi_j \nabla\phi_i \; dx + \int_T (\beta \cdot \nabla\phi_j) \, \phi_i \; dx + \int_T \alpha \, \phi_j \, \phi_i \; dx \right)$$

Allo stesso modo facciamo per il termine noto:

$$(f)_i \quad = \quad \sum_{T \in \mathbb{T}} (f^T)_i \quad = \quad \sum_{T \in \mathbb{T}} \int_T f \, \phi_i \; dx$$

Consideriamo dunque $\forall \; T \in \mathbb{T}$ il problema:

$$\int_T c \, \nabla\phi_j \nabla\phi_i \; dx \; + \; \int_T (\beta \cdot \nabla\phi_j) \, \phi_i \; dx \; + \; \int_T \alpha \, \phi_j \, \phi_i \; dx \quad = \quad \int_T f \, \phi_i \; dx$$

Affrontiamo subito il problema delle condizioni al bordo. Nei vertici di bordo sappiamo già il valore della soluzione u, che corrisponde al valore del dato al bordo g calcolato nel vertice.

Dato che il calcolo diretto viene abbastanza complicato in più di una dimensione, usiamo l'idea di ricondurci ad un elemento di riferimento.

[1] Questo risultato è legato strettamente al fatto che la matrice K_h sia sparsa

Chapter 4

Stime Sugli Errori

4.1 Stime Teoriche Errori

Raccogliamo in questa sezione varie stime per l'errore compiuto nell'approssimazione agli elementi finiti. Inanzitutto ricordiamo il lemma di Cea 2.3.1 per la stima dell'errore compiuto dal metodo Galerkin 2.3.1:

$$|| \ u - u_h \ ||_{\mathbb{V}(\Omega)} \ \leq \ \frac{M}{c} \, Inf_{v_h \in \mathbb{V}_h} \, || \ u - v_h \ ||_{\mathbb{V}(\Omega)}$$

dove M è la costante della continuità e c quella della coercività di $a(\cdot, \cdot)$. Vogliamo però dare delle stime dell'errore compiuto dall'approssimazione agli elementi finiti in varie norme.

4.2 Calcolo Errore $\mathbb{L}^2$

4.3 Calcolo Errore $\mathbb{H}^1$

DA AGGIUNGERE STIME ERRORI ELEMENTI FINITI
FINE SEZIONE

Chapter 5

Codice Matlab

Riportiamo di seguito il codice Matlab in cui si presenta un'implementazione del metodo agli elementi finiti per il problema ellittico 2.1 nel caso particolare di un dominio $\Omega \in \Re^2$. Riportiamo solo il programma principale *fem2.m*, mentre le altre funzioni di supporto utilizzate (come ad esempio per la generazione della mesh, per le formule di quadratura o per i grafici), possono essere trovate online al sito `https://github.com/UniversityProjects/FEM1/tree/master/Project`

Listing 5.1: Elementi Finiti Di Ordine Due Per Il Problema Ellittico

```
%%%%%%%%%%%%%%%%%%%
% FEM for k = 2
%%%%%%%%%%%%%%%%%%%

clc
clear all
close all

% Exatc Solution Flag
  exact_solution = 'yes';
% exact_solution = 'no';


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Mesh Creation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Domain Definition
omega = 'square'; % Unit Square
% omega = 'squareN'; % Unit Square With Neumann Conditions On A Border
% omega = 'igloo'; % Unit Square With Neumann Conditions On Half Border
% omega = 'squareCdisc'; % Disc With Neumann Conditions On Half Border

% Mesh Generator Choice
% meshgen = 'triangle'; % Mesh generated by triangle
  meshgen = 'uniform'; % Uniform mesh

if (strcmp(meshgen, 'triangle'))
    % Build The Triangle Mesh
    disp('--- Building Mesh ---');
    makemesh;

    % Read The Triangle Mesh
    disp('--- Reading Mesh ---');
    readmesh;
```

```matlab
else
    % Uniform Mesh
    disp('--- Building Uniform Mesh ---');
    makeuniform;
end

% List Mesh Details
disp(['       Vertex Number: ' num2str(nver)]);
disp(['       Triangles Number: ' num2str(nele)]);
disp(['       Edge Number: ' num2str(nedge)]);

% Plot The Mesh
disp('--- Drawing Mesh ---');
drawmesh;

% Quadrature Formula
fdq = 'degree=5';

% (xhq, yhq) Quadrature's Nodes
% whq = pesi
disp('--- Quadrature Computation ---');
disp([' quadrature: ', fdq]);
[xhq,yhq,whq] = quadrature(fdq);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Kh Matrix Assembling
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Basis Function Computed On The Quadrature Nodes Of The Riferement
    Element

Nq = length(xhq); % Number Of Quadrature Nodes
phihq = zeros(6,Nq); % Phihq Definition
gphihqx = zeros(6,Nq); % Gradphihq Definition
gphihqy = zeros(6,Nq); % Gradphihq Definition

% Basis Functions Computation Loop
disp('--- Basis Functions Phi Computation ---');
for i=1:6
    for q=1:Nq
        phihq(i,q) = phih2(i,xhq(q),yhq(q));
    end
end

% Basis Functions Gradients Computation Loop
disp('--- Gradient Basis Functions Phi Computation ---');
for i=1:6
    for q=1:Nq
        [gx gy] = gradphih2(i,xhq(q),yhq(q));
        gphihqx(i,q) = gx;
        gphihqy(i,q) = gy;
    end
end

% A Matrix Definition
A = sparse(nver+nedge,nver+nedge);

% b Array Definition
b = zeros(nver+nedge,1);

% Main Computation Loop On Every Triangle
```

```
disp('--- A Matrix and b Array Computation ---');
for iele=1:nele

% Acquire Informations From The iele Elements

    % Vertices Acquisition
    v1 = vertices(iele,1);
    v2 = vertices(iele,2);
    v3 = vertices(iele,3);


    % Vertex 1 Coordinates
    x1 = xv(v1);
    y1 = yv(v1);

    % Vertex 2 Coordinates
    x2 = xv(v2);
    y2 = yv(v2);

    % Vertex 3 Coordinates
    x3 = xv(v3);
    y3 = yv(v3);


 % Jacobian Matrix Computation

    % F Jacobian
    JF = [x2 - x1    x3 - x1
          y2 - y1    y3 - y1];

    % F Jacobian Inverse
    JFI = inv(JF);

    % F Jacobian Inverse Transpost
    JFIT = JFI';


% Single Element Area
    area = 0.5*det(JF);

% KE Matrix Definition
    KE = zeros(6,6);

% Actual Matrix KE Computation Loop
    for i=1:6
        for j=1:i-1 % Loop That Use Matrix Symmetry To Halve The
            Computations
             KE(i,j) = KE(j,i);
        end
        for j = i:6
            for q=1:Nq
                % Image on T (current triangle) Of The Quadrature Node
                % tmp = (xq, yq) = (xhq(q),yhq(q))
                % On The Riferiment Element
                tmp = JF*[xhq(q); yhq(q)] + [x1; y1];
                xq = tmp(1); % Quadrature Node X Coordinate
                yq = tmp(2); % Quadrature Node Y Coordinate
                % Diffusive term (Second Order)
                % c * grad phi(j,q) ** grad phi (i,q) * whq(q)
                diffusive = c(xq,yq)*dot(JFIT*[gphihqx(j,q);...
                                               gphihqy(j,q)],...
                                         JFIT*[gphihqx(i,q);...
```

```
                                                    gphihqy(i,q)]...
                                             )*whq(q);
                        % Reactive Term (First Order)
                        % beta ** grad phi(j,q) * phi (i,q) * whq(q)
                        [b1, b2] = beta(xq,yq);
                        transport = dot([b1; b2], ...
                                       JFIT*[gphihqx(j,q); gphihqy(j,q)]...
                                       )*phihq(i,q)*whq(q);
                        % Transport Term (Zeroth Order)
                        % alpha * phi(j,q) * phi (i,q) * whq(q)
                        reaction = alpha(xq,yq)*(phihq(j,q)*phihq(i,q))*whq(q);
                        % KE(i,j) Sum Update With All Three Terms
                        KE(i,j) = KE(i,j) + diffusive + transport + reaction;
                    end
                    KE(i,j) = 2*area*KE(i,j);
                end
            end

% Recover Triangle's Edges
    l1 = edges(iele,1); % First Edge
    l2 = edges(iele,2); % Second Edge
    l3 = edges(iele,3); % Third Edge

% Global Degrees Of Freedon

    % Vertex i ---> i
    % Edge i   ---> nver
    % This array gives the current triangle's Global Degrees Of Freedom
    dofg = [v1 v2 v3 (nver+l1) (nver+l2) (nver+l3)];


% Global Matrix A Computation
    A(dofg,dofg) = A(dofg,dofg) + KE;


% FE Array Definition
    FE = zeros(6,1);

% Actual Array Fe Computation Loop
    for i=1:6
        for q=1:Nq
            % Image on T (current triangle) Of The Quadrature Node
            % tmp = (xq, yq) = (xhq(q),yhq(q))
            tmp = JF*[xhq(q); yhq(q)] + [x1; y1];
            xq = tmp(1); % Quadrature Node X Coordinate
            yq = tmp(2); % Quadrature Node Y Coordinate
            FE(i) = FE(i) + f(xq,yq)*phihq(i,q)*whq(q);
        end
        FE(i) = 2*area*FE(i);
    end

% Global b Coefficient Computation
    b(dofg) = b(dofg) + FE;


end

% Spy A Matrix
disp('--- Spying A matrix ---');
figure();
spy(A);
```

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Border Conditions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

disp('--- Border Conditions ---');

% Free Nodes Array Definition
NL = [];

% Approximated Solution Array Definition
uh = zeros(nver+nedge,1);

for iv=1:nver
    % Check if the iv vertex is a border vertex
    if (vertexmarker(iv) == 1) % Dirichlet Condition
        uh(iv) = g(xv(iv),yv(iv));
        % Update Constant Term
        b = b - uh(iv)*A(:,iv);
    else % Free Node
        NL = [NL iv];
    end
end


for iedge=1:nedge
    % Border Degree Of Freedom
    dof = nver+iedge;
    % Constant Term Update
    if edgemarker(iedge)==1 % Border Edge
    % Edge Medium Point
        % First Point
        v1 = endpoints(iedge,1);
        x1 = xv(v1);
        y1 = yv(v1);
        % Second Point
        v2 = endpoints(iedge,2);
        x2 = xv(v2);
        y2 = yv(v2);
        % Medium Point Computation
        xm = (x1 + x2) / 2;
        ym = (y1 + y2) / 2;
    % Constant Term Update
        uh(dof) = g(xm,ym);
        b = b -uh(dof)*A(:,dof);
    else % Free Edge
        NL = [NL dof];
    end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Approximate Solution Computation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

disp('--- Solution Computing ---');

% Exctract The True Kh Matrix On The Free Nodes
Kh = A(NL,NL);
```

```
% Exctract The True fh Array On The Free Nodes
fh = b(NL);

% Compute The Approximated Solution
% If iv is a vertex, then uh(iv) is the value
% of uh in that vertex.
% if ie is an edge, uh(nver+ie) is the value
% of uh in the medium point of the edge.
uh(NL) = Kh\fh;

% Solution Plot On Vertices
% disp('--- Drawing Approximated Solution On Vertices---');
% drawuhVer;

% Solution Plot On Edges
disp('--- Drawing Approximated Solution On Edges---');
drawuhEdge;

% Uh Max
disp(['--- Uh max: ' num2str(max(uh)) ' ---']);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Exact Solution Plot
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if (strcmp(exact_solution,'yes'))
    % Exact Solution Plot On Vertices
    % disp('--- Drawing Exact Solution On Vertices---');
    % drawueVer;

    % Exact Solution Plot On Edges
    disp('--- Drawing Exact Solution On Edges---');
    drawueEdge;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% L2 and H1 Error Computation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if (strcmp(exact_solution,'yes'))
    disp('--- L2 and H1 Error Computation ---');

    % Quadrature Formula For Error Computing
    fdq = 'degree=5';
    disp([' quadrature: ', fdq]);

    % (xhq, yhq) Quadrature's Nodes
    % whq = pesi
    [xhq,yhq,whq] = quadrature(fdq);

    % Basis Function Computed On The Quadrature Nodes Of The Riferement
        Element

    Nq = length(xhq); % Number Of Quadrature Nodes
    phihq = zeros(6,Nq); % Phihq Definition
    gphihqx = zeros(6,Nq); % Gradphihq Definition
    gphihqy = zeros(6,Nq); % Gradphihq Definition

    % Basis Functions Computation Loop
```

```
for i=1:6
  for q=1:Nq
        phihq(i,q) = phih2(i,xhq(q),yhq(q));
  end
end

% L2 Error Variable
errL2sq = 0;

% H1 Error Variable
errH1sq = 0;

% Actual Errors Computation
% Works By Computing The Global Errors As A Summation
% Of The Errors On Every Triangle
for iele=1:nele

% Acquire Informations From The iele Elements

    % Vertices Acquisition
    v1 = vertices(iele,1);
    v2 = vertices(iele,2);
    v3 = vertices(iele,3);


    % Vertex 1 Coordinates
    x1 = xv(v1);
    y1 = yv(v1);

    % Vertex 2 Coordinates
    x2 = xv(v2);
    y2 = yv(v2);

    % Vertex 3 Coordinates
    x3 = xv(v3);
    y3 = yv(v3);


% Jacobian Matrix Computation

    % F Jacobian
    JF = [x2-x1    x3-x1
          y2-y1    y3-y1];

    % F Jabobian Inverse
    JFI = inv(JF);

    % F Jacobian Inverse Transposed
    JFIT = JFI';

    % Single Element Area (Triangle's Area)
    area = (1/2)*det(JF);

% Recover Triangle's Edges
    l1 = edges(iele,1); % First Edge
    l2 = edges(iele,2); % Second Edge
    l3 = edges(iele,3); % Third Edge

% Global Degrees Of Freedon

    % Vertex i ---> i
    % Edge i   ---> nver
```

```
        % This row-array holds the current triangle's Global Degrees Of
            Freedom
        dofg = [v1 v2 v3 (nver+l1) (nver+l2) (nver+l3)];

    % Recover the uT coefficients
        uT = uh(dofg);

        % Tmp variables to hold the element result
        sqL2 = 0;
        sqH1 = [ 0; 0];
        normsqH1 = 0;

        % Computation Over Weighting Nodes
        for q=1:Nq
            % Compute the sum on phi(i)
            tmpL2_1 = 0;
            tmpH1 = [0; 0];
            for i=1:6
                tmpL2_1 = tmpL2_1 + uT(i)*phihq(i,q);
                tmpH1 = tmpH1 + JFIT*uT(i)*[gphihqx(i,q); gphihqy(i,q)];
            end
            % Error Computation
            tmpL2_2 = JF*[xhq(q);yhq(q)] + [x1;y1];
            xq = tmpL2_2(1);
            yq = tmpL2_2(2);
            sqL2 = sqL2 + (ue(xq,yq) - tmpL2_1)^2 * whq(q);
            %sqH1 = sqH1 + [ux(xq,yq); uy(xq,yq)] - tmpH1;
            sqH1 = sqH1 + ue(xq,yq) - tmpH1;
            normsqH1 = normsqH1 + dot(sqH1, sqH1)*whq(q);
        end

        sqL2 = 2*area*sqL2;
        normsqH1 = 2*area*normsqH1;

        % L2 Error On The Element (Squared)
        errL2sq = errL2sq + sqL2;

        % H1 Error On The Element (Squared)
        % ErrH1 = errL2(u) + errL2(grad u)
        errH1sq = errH1sq + errL2sq + normsqH1;

    end

    % Final L2 Error Computation
    errL2 = sqrt(errL2sq);

    % Final H1 Error Computation
    errH1 = sqrt(errH1sq);

    disp(['       L2 Error: ' num2str(errL2)]);
    disp(['       H1 Error: ' num2str(errH1)]);


end
```

Chapter 6

Esperimenti Numerici

Consideriamo in questo capitolo un confronto tra soluzione esatta e soluzione approssimata per verificare a livello pratico i risultati teorici di convergenza rispetto al parametro geometrico h della specifica mesh.

Index

Bibliography

[Bab71] Ivo Babuska. Error-bounds for finite element method. *Numerische Mathematik*, 16:322–333, 1971.

[Bre10] Haim Brezis. *Functional Analysis, Sobolev Spaces And Partial Differential Equations*. Springer, 2010.

[BS94] Ivo Babuška and Manil Suri. The p and h-p versions of the finite element method, basic principles and properties. *SIAM review*, 36(4):578–632, 1994.

[BS07] Susanne C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Method*. Springer, third edition, 2007.

[BSK81] Ivo Babuška, Barna A Szabo, and I Norman Katz. The p-version of the finite element method. *SIAM journal on numerical analysis*, 18(3):515–545, 1981.

[CHB09] J. Austin Cottrell, Thomas J.R. Hughes, and Yuri Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. Wiley, 2009.

[Cia78] P. G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland, 1978.

[Eva90] Lawrence C. Evans. *An Introduction To Partial Differential Equations*. American Mathematical Society, 1990.

[FP13] Davide Luigi Ferrario and Renzo Piccinini. *Simplicial Structures In Topology*. Springer, 2013.

[Jac98] Jhon David Jackson. *Classical Electrodynamics*. Jhon Wiley and Sons, third edition, 1998.

[Len86] M Lenoir. Optimal isoparametric finite elements and error estimates for domains involving curved boundaries. *SIAM Journal on Numerical Analysis*, 23(3):562–580, 1986.

[Qua08] Quarteroni. *Modellistica Numerica Per Problemi Differenziali [Trad: Numerical Modelling For Differential Problems]*. Springer, fourth edition, 2008.

[Saa03] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, second edition, 2003.

[Sal10] Sandro Salsa. *Equazioni Alle Derivate Parziali: metodi, modelli e applicazioni [Trad: Partial Differential Equations: methods, models and applications]*. Springer, second edition, 2010.