Sede Amministrativa: Universita‘ degli Studi di Padova

Dipartimento di Matematica Pura ed Applicata

DOTTORATO DI RICERCA IN: MATEMATICA COMPUTAZIONALE
XII CICLO

Adaptivity in the Finite Element Method

Coordinatore: Ch.mo Prof. Renato Zanovello
Supervisore: Ch.mo Prof. Maria Morandi Cecchi

Dottorando: Fabio Marcuzzi

31 dicembre 1999

Ringraziamenti

Desidero ringraziare vivamente la Professoressa Maria Morandi Cecchi per i numerosi consigli, la generosa disponibilita‘ alla discussione degli argomenti, la liberta‘ e la fiducia accordatemi nello svolgimento del lavoro di tesi e per il gradito coinvolgimento nelle attivita‘ di ricerca in generale.

Ringrazio anche il Coordinatore del dottorato, Prof. Renato Zanovello.

Desidero inoltre ringraziare il Professor Thomas J.R. Hughes di Stanford per le importanti discussioni e spiegazioni da lui ricevute, nonche’ per la disponibilita’ e la gentile accoglienza nel suo gruppo di ricerca ed all’utilizzo delle risorse di Stanford.

Desidero ringraziare inoltre il Professor Gene Golub di Stanford, per la sua grande cordialita’, per il suo insegnamento e per i preziosi consigli da lui ricevuti.

Il mio ringraziamento va anche a tutte le persone che, durante questi due anni e mezzo di lavoro, mi hanno direttamente aiutato o indirettamente, grazie al loro impegno nel lavoro, al Dipartimento di Matematica Pura ed Applicata dell’Universita‘ di Padova ed in particolare al personale della Biblioteca.

Ringrazio i miei genitori Marirosa ed Elio ed i miei nonni Rina e Davino per l’aiuto che mi hanno dato e ringrazio moltissimo i miei suoceri Gina e Roberto per il loro grande supporto alla mia famiglia durante tutto questo periodo.

Infine, ma con grande affetto, ringrazio mia moglie Antonella per aver accettato i disagi derivanti dalla mia scelta ed il mio piccolo Alberto per l’enorme gioia che ci sta portando in famiglia. A loro e‘ dedicato il mio lavoro.

Introduzione

In questa tesi viene affrontato con una visione globale il problema dell'Adattivita' nel metodo degli Elementi Finiti. Sono stati studiati i singoli aspetti del problema, ed e' stata data anche grande importanza alle interrelazioni che eventualmente esistano tra i vari aspetti.

L'importanza degli algoritmi adattativi per la risoluzione di problemi alle derivate parziali, settore dove sono nati e vengono maggiormente applicati gli Elementi Finiti, e' ormai riconosciuta universalmente per i vantaggi pratici che essa offre, nonostante le soluzioni esistenti non siano ancora completamente soddisfacenti.

Da un punto di vista pratico, infatti, l'adattivita' permette un uso ottimizzato delle risorse di calcolo ed e' dunque un contributo importante della matematica alla risoluzione di problemi reali che per la complessita' e l'accuratezza richieste oggigiorno non possono essere risolti solo grazie alla potenza delle risorse di calcolo (hardware).

L'uso dell'adattivita' negli elementi finiti e' trasversale tra le applicazioni e largamente applicabile. Benche' non ci sia un unico algoritmo adattativo che possa essere applicato a tutti i problemi differenziali alle derivate parziali (in verita' e' quasi vero il contrario), ci sono molte questioni di base nell'adattivita', che possono essere risolte indipendentemente dal problema considerato.

La formulazione generale di un algoritmo adattativo e' la seguente [100]:

1. si costruisce una discretizzazione (*mesh*) grezza iniziale T_0, di elementi finiti, che rappresenti sufficientemente bene la geometria del problema; si pone $k = 0$;

2. si risolve il problema discreto su T_k;

3. se l'errore di approssimazione globale stimato e' sufficientemente piccolo l'algoritmo termina, altrimenti si decide quali elementi devono essere raffinati e si costruisce la mesh successiva T_{k+1}; si sostituisce k con $k + 1$ e si ritorna al passo 2;

Per problemi dipendenti dal tempo, alcune modifiche sono necessarie:

1. la stima a-posteriori dell'errore deve essere ripetuta dopo qualche passo temporale;

2. il procedimento di raffinamento nello spazio deve essere accompagnato da un controllo del passo temporale;

3. in generale e' necessario aggiungere un meccanismo di de-raffinamento della mesh;

4. talvolta e' anche necessario ricostruire la mesh daccapo.

I tre sotto-problemi principali, intimamente correlati, dell'adattivita' sono: stima a-posteriori dell'errore di approssimazione, generazione ed adattamento della mesh, soluzione del sistema lineare di equazioni. L'approccio utilizzato per risolvere uno di essi pu influenzare fortemente gli approcci e/o le prestazioni che possono essere ottenute per risolvere gli altri. In questa tesi i tre argomenti vengono presentati separatamente, dato che rappresentano tre settori computazionali piuttosto differenti, pero' vengono messe in evidenza diverse sinergie tra di essi.

In un'analisi agli elementi finiti e' in generale preferibile ottenere un'approssimazione uniforme della soluzione vera in tutto il dominio del problema. Generalmente, la soluzione di un problema reale alle derivate parziali presenta differenti livelli di regolarita' in regioni differenti del dominio e dunque per averne un'approssimazione uniforme con gli elementi finiti e' necessario distribuire questi ultimi in modo non-uniforme e tale da adattarsi alle caratteristiche della soluzione. L'*adattivita'* nel metodo degli Elementi Finiti e' la soluzione a questo problema. Una distribuzione dappertutto uniforme di elementi (cioe' una mesh uniformemente raffinata) corrisponde, invece, generalmente ad un considerevole spreco delle risorse di calcolo.

L'adattamento della mesh puo' essere effettuato generalmente in due modi: costruendo daccapo la mesh con criteri modificati, oppure modificando la mesh esistente. In questa tesi viene considerato il secondo approccio e si parla di *raffinamento* quando vengono aggiunti elementi e *de-raffinamento* quando alcuni elementi vengono rimossi. Benche' alcuni metodi qui presentati possano essere estesi anche al caso non-lineare, nel seguito si suppone che l'operatore differenziale oggetto della discretizzazione agli elementi finiti sia lineare.

Il Capitolo 1 inizia con l'esposizione di un aspetto fondamentale del metodo degli Elementi Finiti e cioe' la famiglia dei Metodi dei Residui Pesati: la soluzione approssimante ottenuta applicando questi metodi e' quella che rende nullo il prodotto interno tra il residuo della forma forte del problema differenziale, calcolato sulla soluzione approssimante stessa, e lo spazio di funzioni-peso prescelto. In particolare viene descritto il metodo di Galerkin, molto utilizzato negli elementi finiti perche' fornisce la miglior approssimazione nella norma dell'energia, la quale e' fondamentale in certe applicazioni, ad esempio nei problemi di elasticita' lineare.

E' contributo originale di questo Capitolo il metodo analitico di scelta dello spazio delle funzioni peso che rende minimo l'errore di approssimazione commesso dalla soluzione calcolata (appartenente ad uno spazio a dimensione finita) rispetto alla soluzione vera (che appartiene ad uno spazio a dimensione infinita), nella norma indotta dal prodotto interno che e' stato utilizzato per la formulazione del metodo dei residui pesati (Teorema 1.1). Per il metodo degli Elementi Finiti il prodotto interno utilizzato e' generalmente quello in L_2. Questo metodo analitico e' utile, ad esempio, per la risoluzione di problemi ai limiti in cui la forma bi-lineare della corrispondente formulazione debole del problema non e' simmetrica e per i quali dunque il metodo di Galerkin non fornisce piu' la miglior approssimazione nella norma dell'energia. Il caso considerato come esempio applicativo e' quello relativo all'equazione di convezione-diffusione stazionaria, per la quale e' ben noto che, nel caso di convezione dominante sulla diffusione, il metodo di Galerkin fornisce una soluzione generalmente instabile.

Nel Capitolo 2 vengono dapprima descritti i principali metodi di stima a-posteriori dell'errore di approssimazione utilizzati nei metodi adattativi agli elementi finiti. Il contributo originale consiste nella formulazione di un nuovo stimatore di tipo implicito che consiste nel risolvere, per ogni sottodominio in cui e' richiesta la stima, un problema di aggiornamento (globale) della soluzione in seguito ad un raffinamento della discretizzazione nel sotto-dominio stesso. Il suo utilizzo e' giustificato dal presupposto, verificato sperimentalmente, che il costo computazionale della risoluzione di tali problemi di aggiornamento e' accettabile, ammesso che venga utilizzato un opportuno solutore di sistemi di equazioni lineari, poiche' il numero di variabili che e' realmente necessario aggiornare, per ogni sottoproblema, e' piccolo rispetto al totale. Nella Parte III della Tesi vengono presentati dei metodi di soluzione di sistemi lineari che aggiornano solo il sottoinsieme di variabili necessario (Capitolo 5).

I punti di forza di questo stimatore sono la sua semplicita' ed il fatto di stimare l'errore di approssimazione globale determinato dalla discretizzazione in una zona locale della mesh, e dunque di stimare anche l'errore procurato al di fuori della zona stessa. Questo, il cosiddetto *pollution error*, non e' riconosciuto dagli stimatori a-posteriori esistenti, poiche' questi sono basati su elaborazioni locali. Nella pratica avviene che la presenza di questo tipo di errore ne alteri le stime. Esso e' presente, ad esempio, nelle soluzioni agli elementi finiti di problemi su domini poligonali con angoli rientranti o con discontinuita' nelle condizioni al contorno e dunque nella maggior parte delle applicazioni reali.

Il Capitolo 3 comincia con una panoramica dei metodi agli elementi finiti per problemi non-stazionari e delle corrispondenti formulazioni adattative, evidenziando tra l'altro l'interazione che esiste tra stime dell'errore nello spazio ed errori di discretizzazione nel tempo e tra stime dell'errore nel tempo ed errori di discretizzazione nello spazio. Viene anche presentata una formulazione adattativa originale del metodo di semi-discretizzazione di tipo Taylor-Galerkin applicato ad un problema di acque basse. Il contributo successivo consiste nel proporre uno stimatore unico per gli elementi finiti spazio-tempo, che distingue la componente di errore nello spazio dalla componente di errore nel tempo. Questa proprieta' permette di:

- attenuare (se non eliminare) l'interazione accennata qui sopra;
- se utilizzata insieme al meccanismo di raffinamento direzionale presentato al Capitolo 4,permette di costruire una mesh di elementi finiti spazio-tempo non-strutturata piu' efficiente di quella che si avrebbe utilizzando due stimatori separati (come usualmente viene fatto).

Nel Capitolo 4 vengono presentate alcune tecniche di raffinamento/de-raffinamento per mesh di elementi finiti nello spazio e nello spazio-tempo. E‘ contributo originale un algoritmo ricorsivo di triangolazione che e‘ adatto a descrivere problemi in cui ci sono parametri che variano in modo arbitrario (cio‘ viene fatto posizionando i nodi sulle interfacce, se presenti, o comunque in modo da garantire che dentro all'elemento i parametri siano costanti a meno di una tolleranza prefissata) e consente di costruire mesh adattative non-strutturate (in quanto vi e‘ localmente flessibilita‘ sufficiente nella costruzione della mesh in modo da non imporre nessuna strutturazione locale). L'algoritmo viene presentato per mesh di triangoli ma puo‘ essere esteso ad altri tipi di elemento finito bi- e tridimensionali.

Questo algoritmo produce una discretizzazione che ha proprieta‘ interessanti per la risoluzione del sistema di equazioni lineari risultante dall'applicazione del metodo degli elementi finiti ed utilizzando la mesh generata. In particolare, se il sistema viene risolto con un metodo diretto e viene utilizzato come ordine di eliminazione quello corrispondente alla gerarchia ricorsiva della mesh (percorsa nel senso inverso di quello di costruzione della mesh):

1- il fill-in generato durante la fattorizzazione della matrice rimane confinato all'interno dei singoli triangoli che compongono la mesh di livello-0 (Teorema 4.1);

2- l'albero di eliminazione (*elimination-tree*) risulta determinato per costruzione (Teorema 4.2).

Sperimentalmente si nota la formazione di un livello soddisfacentemente basso di fill-in sia per il problema di eliminazione dei soli nodi interni alla triangolazione di livello-0, qui chiamato problema di omogeneizzazione, che per la fattorizzazione totale della matrice. Inoltre, e‘ da notare che il fatto di seguire per l'eliminazione l'ordinamento determinato per costruzione permette di evitare di dover eseguire un algoritmo di pre-ordinamento, che generalmente richiede un tempo di esecuzione pari al $30 - 50\%$ del tempo richiesto per la risoluzione del sistema.

La Parte III della tesi presenta la caratterizzazione del problema di aggiornamento della soluzione dopo un passo di raffinamento/de-raffinamento della discretizzazione come il problema di risolvere un sistema perturbato con membro destro uguale a zero tranne che in pochi elementi. Il Capitolo 5 propone due etsensioni originali ai *metodi diretti* per risolvere questo problema aggiornando solo le variabili il cui valore dell'aggiornamento supera una soglia prefissata di accuratezza.

Il primo metodo si chiama Waves-Bordering-approssimato e risolve il sistema perturbato partendo da un insieme iniziale di equazioni ed allargandolo finche‘ non ha aggiornato tutte le variabili necessarie affinche‘ il residuo sia inferiore ad una soglia prefissata. I risultati sperimentali mostrano una certa capacita‘ dell'algoritmo a sfruttare la localita‘ dell'aggiornamento.

Il secondo metodo si chiama GreenLU e determina a-priori l'insieme di variabili da aggiornare, su cui poi applica un metodo diretto standard. L'insieme di variabili viene determinato facendo uso delle funzioni di Green e del principio di sovrapposizione. I risultati sperimentali mostrano una certa capacita‘ dell’algoritmo a non sovra-stimare troppo l’insieme di variabili da aggiornare, e dunque a sfruttare gran parte del risparmio computazionale possibile rispetto all’aggiornamento di tutte le variabili del sistema.

Index

Introduction

In this Thesis, the problem of *adaptivity* in the Finite Element Method is considered with a global perspective. This means that other than studying the single aspects of the problem, also a big importance has been given to the interrelations existing between them.

The importance of adaptive numerical methods for the solution of initial/boundary value problems involving partial differential operators, that is the field where the Finite Element method was born and has been mainly applied, it is universally recognized because of the advantages that it offers in practical computations. Moreover, the existing adaptive methods are generally not considered to be sufficiently mature and there is a big expectation of future results and even better performances.

From a practical point-of-view, an adaptive algorithm allows an optimized use of the computing resources and represents an important contribution from the mathematics to the solution of those real problems which, due to their complexity and to the accuracy required today, would exceed in general the computing power available.

The topic of *adaptivity* is transversal and pervasive in the application of the finite element method. In fact, although there is no universal adaptive algorithm that can be applied to all partial differential problems (indeed it is quite true the opposite), there are many basic issues in adaptivity that holds for all problems. Hence, general formulations and algorithmic strategies can be developed.

The general formulation for an adaptive algorithm is the following [100]:

1. construct an initial coarse mesh T_0 representing sufficiently well the geometry of the problem; put $k = 0$;

2. solve the discrete problem on T_k;

3. for each element of T_k compute the a-posteriori error estimate;

4. if the estimated global error is sufficiently small then stop the algorithm, otherwise decide which elements must be refined and construct the next mesh T_{k+1}; replace k by $k+1$ and return to step 2;

For time-dependent problems, some changes have to be made:

1. the accuracy of the computed numerical solution has to be estimated every few time-steps;

2. the refinement process in space should be coupled with a time-step control;

3. a partial coarsening (i.e. un-refinement) of the mesh might be necessary;

4. occasionally, a complete re-meshing could be desirable.

A-posteriori estimation of the approximation error, mesh generation and adaptation, solution of linear systems of equations, are the three main and intimately related issues in adaptivity. This means that the approaches used for one of them can influence strongly the approaches or the performances that can be obtained for the others. In this thesis they are presented separately, since they represent three quite different fields of numerical analysis, but all the interrelations between them are put on evidence.

In a Finite Elements Analysis it is desirable to have a uniform approximation of the true solution throughout the whole problem-domain. In general, a real-problem solution presents different degrees of smoothness in different regions of the domain, hence to approximate it uniformly it is necessary to distribute the elements non-uniformly and in such a way that their distribution adapts to the characteristics of the solution.

Adaptivity in Finite-Elements Analysis is the solution to this problem. It is to be noted that a uniform high density of finite-elements everywhere (i.e. a *uniformly refined* mesh), it is in general a computational waste.

Finite-elements mesh adaptation can be performed substantially in two ways, building a completely new mesh or modifying the existing one. In this thesis we consider the second approach and we speak of *refinement* when elements are added and of *un-refinement* when elements are removed. The thesis is subdivided in three parts. Altough some methods here presented can be extended also to the non-linear case, in the following we will assume that the differential operator discretized using finite elements is linear.

Chapter 1 begins with the exposition of a fundamental ingredient of the finite element method, i.e. the concept of Weighted Residuals Method. The solution computed by applying this method has the property that it gives a null inner-product (i.e. an orthogonality condition) between the corresponding *residual* of the strong form of the differential problem and the space of weighting functions adopted. A different choice of the weighting-functions space gives a different weighted residuals method. A well known choice corresponds to the Galerkin method, very often used in the finite element method since it gives a solution which is the best approximation in the *energy-norm*, a fundamental property in some applications, e.g. linear elastostatics.

An original contribution is the analytic method for choosing the weighting-functions space in such a way that the solution obtained is the best approximation in the norm induced by the inner-product which has been used for the formulation of the weighted-residuals method (in the finite element method this is generally the L_2-norm). This method has been named the Best-Approximation Weighted-Residuals method (BAWR). It is useful, e.g., to solve initial/boundary value problems whose variational (i.e. weak) formulation presents an unsymmetric bilinear form, because for these problems the Galerkin method looses its optimality in the energy-norm. The practical example here considered is that of the steady advection-diffusion equation, for which it is well known that, in case of the advection dominates over the diffusion, the Galerkin method produces a solution even unstable.

In Chapter 2 are described the main methods used in the adaptive finite elements for obtaining a-posteriori estimates of the approximation error. An original contribution consists in the formulation of a new estimator, which obtains the a-posteriori error estimate by solving a series of global solution-update problems, each of them corresponding to a local refinement into a different subdomain. The computational cost for this estimator, which would be in priciple prohibitive, is instead acceptable if a proper linear-systems solver is used, since the experimental evidence (which has also a theoretical justification, see Chapter 5) shows that the number of variables in the linerar system which must be updated for each solution-update problem is a low percentage of the total. In the third Part of this thesis, some methods which update only the required subset of variables are presented.

The advantages offered by this estimator are its simplicity and the fact of estimating directly the approximation error created in the overall domain by the discretization in a local subdomain, and hence also outside the subdomain itself. This approximation error, the so-called *pollution error*, cannot be recognized by existing a-posteriori error estimators because they are based on local computations. For these estimators, it turns out in practice that the pollution error alters in general the values of the a-posteriori estimates. It is present, for example, in all finite element solutions of problems solved on domains with polygonal domains with re-entrant corners and/or discontinuities on boundary conditions, and therefore on the majority of real applications.

Chapter 3 begins with an overview of the finite element methods for time-dependent problems and of the corresponding adaptive formulations. The interactions existing between a-posteriori error estimates inspace and approximation error in time (and vice-versa, between a-posteriori error estimates in time and the approximation error in space) are put on evidence. It is also presented an original adaptive formulation of the Taylor-Galerkin semi-discretization method applied to a shallow waters problem.

An original contribution is the proposal of a single space-time adaptive scheme for space-time finite elements, which is based on an a-posteriori estimator capable of distinguishing between the error component in time and the error component in space. This property, permits of:

- diminishing (if not eliminating) the interaction previously mentioned;
- building an unstructured mesh of space-time finite elements more efficient than that would be obtained using the techniques present in the literature. This can be achieved if the refinement is made *directionally*, as explained in Chapter 4.

In Chapter 4, some refinement/un-refinement techniques for finite elements in space and in space-time are briefly presented. An original contribution is a ricursive triangulation algorithm which is suited for problems with arbitrarily varying coefficients (this is done by positioning the nodes at the interfaces, if present, or anyway so that within each element the parameters are constant within a predetermined tolerance) and which permits the construction of unstructured adaptive meshes (since there is sufficient local flexibility in placing new nodes) The algorithm is presented for meshes of triangles, but it can be extended to other types of bi- and tri-dimensional elements.

This algorithm produces a discretization which has interesting properties for the solution of the linear system of equations arising from the application of the finite element method on this discretization. In particular, if the system is solved using a *direct method* and the elimination ordering adopted corresponds to following backward the hierarchy of nodes generated by the recursive algorithm, it follows that:

- the *fill-in* generated during matrix factorization remains confined within each triangle which compose the 0-level (i.e. initial) mesh;
- the *elimination-tree* is determined auomatically by construction.

The experimental results (i.e. comparison with the best pre-ordering algorithm available) indicate a low level of fill-in produced both for the problem of eliminating only the nodes internal to the 0-level mesh (here called the *numerical homogenization* problem) and for the problem of the total factorization of the matrix. Moreover, here it can be avoided the execution of a pre-ordering algorithm (because the elimination ordering is determined by construction), which generally requires an execution time between 30% and 50% of the time needed then for factorizing the matrix.

Part III of the thesis starts from a characterization of the solution-update problem after a mesh refinement/un-refinement step as the problem of solving a perturbed linear system with an almost null right-hand-side. Then, in Chapter five, two original solution methods that can be embedded in a direct solver, are presented. Both methods are designed to factorize only the submatrix corresponding to the variables whose update will be higher than a prescribed threshold.

The first method has been named *Approximate Waves-Bordering* and solves the perturbed linear system starting from an initial set of equations and enlarging it until have been factored all the equations required to have a *residual* below a predetermined threshold. The experimental results show a certain capability of the algorithm to exploit the locality of the solution-update.

The second method has been named *GreenLU* and determines a-priori the subset of variables to be updated, upon which it is then applied a standard direct method. This subset of variables is determined making use of the Green's functions and of the Superposition

Principle. The experimental results show a certain capability of the algorithm to not overestimate too much the subset of variables to be updated, and hence to save a big part of the computational cost that it is possible to save instead of updating of all system's variables.

PART I

The Approximation Error and its A-Posteriori Estimation

Chapter 1

1

Best Approximations using Weighted-Residuals Methods

The topic here discussed is a premise for doing adaptivity efficiently: to approximate at best the solution using the available degrees of freedom. It is an optimization of the discretization algorithm.

1.1 The Approximation Error in the Finite-Element Method for Elliptic Problems

This is an introductory Section. Here it is briefly discussed the approximation scheme which is used by the Finite-Element method, and the resulting approximation error properties. A complete presentation of the computational and mathematical aspects involved in the Finite-Element method can be found e.g. in [97] [4] [73] [80] [41] [21]

Let us start with a presentation of an abstract variational approximation theory, useful for the solution of elliptic boundary value problems. Given an Hilbert space V on R with its inner product and the induced norm $||\cdot||_V$, the bilinear form $a(\cdot,\cdot)$ on $V \times V$ is said to be *continuous* if:

$$\exists M \quad \text{such that} \quad \forall u \in V, \forall v \in V, a(u,v) \leq M||u||_V||v||_V \tag{1.1}$$

and *V-elliptic* if:

$$\exists \alpha > 0 \quad \text{such that} \quad \forall v \in V, a(v,v) \geq \alpha||v||_V^2 \tag{1.2}$$

Let be given also a linear form $L(\cdot)$ continuous on V, i.e. L is an element of the topological dual V' of V in which is defined the dual norm:

$$||L||_{V'} = \sup_{v \in V, ||v||_V = 1} L(v) \qquad . \tag{1.3}$$

With these hypotheses, the well-known Lemma due to Lax and Milgram (for a demonstration see e.g. [41] [90]) states that there exists a unique solution $u \in V$ to the abstract variational problem:

$$a(u,v) = L(v) \qquad \forall v \in V \tag{1.4}$$

and also that the linear application from L to u is continuous from V' to V.

The importance of the formulation (1.4) will be clear in the following. Now, the connection between this problem and a boundary-value problem is shown. Let Ω be an open, bounded subset of R^n with boundary $\partial\Omega$ and V be a closed subspace of $H^1(\Omega)$, a *Sobolev space*, such that

$$H_0^1(\Omega) \subseteq V \subseteq H^1(\Omega) \tag{1.5}$$

(in this way V is an Hilbert space with the norm induced by $H^1(\Omega)$). Let us define the partial differential operator:

$$Au = -\sum_{i,j=1}^{n} \frac{\partial}{\partial x_i}\left(a_{ij}\frac{\partial u}{\partial x_j}\right) + a_0 u \tag{1.6}$$

and introduce the class of elliptic, second-order boundary-value problems, which is often used in applications: find u such that

$$Au = f \tag{1.7}$$

where u is a function sufficiently regular, e.g. $u \in H^2(\Omega)$, $f \in L^2(\Omega)$ is a given function, the coefficients a_{ij} and a_0 are bounded and measurable functions in Ω which verify the *ellipticity conditions*:

$$\exists\alpha > 0 \text{ such that } \forall\psi \in R^n \quad \sum_{i,j=1}^{n} a_{ij}(x)\psi_j\psi_i \geq \alpha|\psi|^2 \text{ a.e. in } \Omega \tag{1.8}$$

$$\exists\alpha_0 \text{ such that } \forall x \in \Omega \quad a_0(x) \geq \alpha_0 \text{ a.e. in } \Omega \tag{1.9}$$

and proper boundary conditions of the Dirichlet and Neumann type are applied:

$$u = u_0 \qquad on \quad \Gamma_D \tag{1.10}$$

$$\frac{\partial u}{\partial n} = g \qquad on \quad \Gamma_N \tag{1.11}$$

where $\dfrac{\partial u}{\partial n}$ is in general the co-normal derivative, $\Gamma_D \cup \Gamma_N = \partial\Omega$ and $\Gamma_D \cap \Gamma_N = 0$, $g \in L^2(\Omega)$ and $u_0 \in H^1(\Omega)$ are given functions. The following facts can be demonstrated (see e.g. [90]). Said

$$V = \{v \in H^1(\Omega); \quad v = 0 \quad on \quad \Gamma_D\} \tag{1.12}$$

if we take $u \in H^1$ and multiply both sides of (1.7) by $v \in V$ and integrate on Ω, i.e.

$$\int_\Omega vAu \quad d\Omega = \int_\Omega vf \quad d\Omega \qquad \forall v \in V \tag{1.13}$$

and we apply the well-known *Green's formula*, after defining $y = u - u_0$ (hence $y \in V$ by (1.10)), we obtain the following problem: find $y \in V$ such that $\forall v \in V$ it holds

$$a(y, v) = \int_\Omega v f d\Omega - a(u_0, v) + \int_{\Gamma_N} v g d\sigma \tag{1.14}$$

where $a(\cdot, \cdot)$ is defined on $V \times V$ by:

$$a(y, v) = \int_\Omega \left(\sum_{i,j=1}^{n} a_{ij} \frac{\partial y}{\partial x_j} \frac{\partial v}{\partial x_i} + a_0 y v \right) d\Omega \tag{1.15}$$

Now, if we define:

$$L(v) = \int_\Omega v f d\Omega - a(u_0, v) + \int_{\Gamma_N} v g d\sigma \tag{1.16}$$

we see that (1.14) is a problem of the type (1.4). It is easy to see in this case that:

$$\forall v \in V \quad a(v, v) \geq \alpha |v|^2_{1,\Omega} + \alpha_0 ||v||^2_{0,\Omega} \tag{1.17}$$

hence the bilinear form $a(\cdot, \cdot)$ defined in (4) is *V-elliptic* at least when $\alpha > 0$. Then there exists a unique solution $y \in V$ which solves the problem (1.14), and the function $u = y + u_0$, $u \in H^1$ satisfies the boundary-value problem (1.7) (1.10) (1.11) in the sense of distributions, i.e. it is a *weak solution* [95].

To numerically approximate the solution of this problem let us take a finite-dimensional subspace V_h, which is an Hilbert space contained in V with the norm of V, and solve the approximate problem: find $u_h \in V_h$ such that

$$a(u_h, v_h) = L(v_h) \qquad \forall v_h \in V_h \tag{1.18}$$

which has also a unique solution for the same Lax-Milgram's Lemma. In the Finite Element method, V_h is chosen as a space of *finite elements* [41] and h is a parameter representative of the maximal dimension of the finite elements used in the discretization V_h. Be $||\cdot||_V$ the norm in V, for the approximate problem (1.18) it is available a fundamental abstract error estimate, contained in the following Lemma [41]:

Cea's Lemma: "There exists a constant C independent of the subspace V_h such that

$$||u - u_h||_V \quad \leq \quad C \quad \inf_{v_h \in V_h} ||u - v_h||_V \tag{1.19}$$

Consequently, a sufficient condition for convergence is that there exists a family (V_h) of subspaces of the space V such that, for each $u \in V$,

$$\lim_{h \to 0} \inf_{v_h \in V_h} ||u - v_h|| = 0 \tag{1.20}$$

Proof: Let w_h be an arbitrary element in V_h. It follows from (1.4) and (1.18) that $a(u - u_h, w_h) = 0$. Using the same constants α and M as in (1.2) (1.1), we have, for any $v_h \in V_h$

$$\alpha||u - u_h||^2 \leq a(u - u_h, u - u_h) = a(u - u_h, u - v_h) \leq M||u - u_h||||v - v_h||$$

and the conclusion follows with $C = M/\alpha$.

This simple result is in part responsible for the great success of the variational approach: with this result, the problem of determining the error is thus conducted to a well-defined problem in approximation theory: to evaluate the quantity $\inf_{v_h \in V_h} ||u - v_h||$, i.e. the distance in V between the solution u and the subspace $V_h \subset V$.

Assuming appropriate smoothness on the function u, it can be shown [41] that this distance is itself bounded by a constant times h^β, for some exponent $\beta > 0$. Usually this majorization is given by means of the interpolation error (to determine whose bounds a certain number of results from approximation theory in Sobolev spaces are available [41]). Hence, this bounds give the additional information that, for a smooth enough solution u, there exists a constant $C(u)$ independent of h such that

$$||u - u_h|| \leq C(u)h^\beta \tag{1.21}$$

In this case it is said that the order of convergence as $h \to 0$ is β, i.e.

$$||u - u_h|| = O(h^\beta) \tag{1.22}$$

The formulation (1.13) suggests us that, starting from the original differential problem, the variational formulation may be obtained by multiplying both sides of the differential equation by an arbitrary weighting-function $w_h \in W_h$, integrating over all the domain and then applying the Green's formula. The fact of multiplying by a weighting-functions space and equating the integrals gives to this approximation method also an intepretation as a *weighted-residuals method* [105], i.e. the orthogonality between the residuals of the strong form of the differential problem and the space of the weighting-functions is imposed:

$$(w_h, f - \mathcal{L}u_h) = (w_h, \mathcal{L}(u - u_h)) = 0 \qquad \forall w_h \in W_h \tag{1.23}$$

A different choice of the weighting functions space leads to a different weighted-residuals method and the resulting characteristics of the approximation error are also different. If we choose $W_h = V_h$ (as was done in (1.13)) we obtain the *Galerkin method* and the resulting properties of the approximation error are presented at the next Section. If W_h is chosen as a suitable space formed by Dirac distributions, we obtain the *Collocation method*, i.e. the solution u_h is that which force the residuals to vanish at a prescribed finite set of points in the domain Ω.

Obviously, the number of possible weighted-residuals methods is, at least in principle, infinite. The aim of this Chapter is to present some analytic methods to choose the weighting-functions spaces that bring to the best-approximation in some norm of practical interest. In the following Section we will start with the well-known Galerkin method. Then, in Section 1.4, an original method will be presented.

1.2 The Galerkin Weighted-Residuals Method

The Galerkin method is the most common choice adopted by the Finite Element users (and was adopted also in the general presentation made in Section 1.1). Its choice for the weighting-functions space takes functions from the same space adopted for the approximating function u_h (e.g. H^1 in (1.13)) and requires that they satisfy the homogeneous counterpart of the Dirichlet boundary conditions applied (e.g. it takes V as in (1.12)).

When the bilinear form $a(\cdot,\cdot)$ defined in $V \times V$ is *symmetric* (i.e. $a(u,v) = a(v,u) \quad \forall u, v \in V$), there is a remarkable interpretation of the discrete solution resulting from the application of the Galerkin method. Since in this case $a(\cdot,\cdot)$ is an inner product and, subtracting (1.18) from (1.4), we have

$$a(u - u_h, v_h) = 0 \quad \forall v_h \in V_h \qquad , \tag{1.24}$$

it follows that u_h is the projection over V_h of the exact solution u, with respect to the inner-product $a(\cdot,\cdot)$. We have, in this case, the following optimality condition:

$$a(u - u_h, u - u_h) = \inf_{v_h \in V_h} a(u - v_h, u - v_h) \tag{1.25}$$

For example, if the differential operator is the Laplacian, the solution obtained applying the Galerkin method is the mean-square projection of the gradient of the exact solution [21]. The norm induced by the inner-product $a(\cdot,\cdot)$, i.e.

$$||v||_E = \sqrt{a(v,v)} \tag{1.26}$$

is called the *energy*-norm.

Moreover, when the bilinear form $a(\cdot,\cdot)$ is symmetric, the solution u of the problem (1.4) is coincident with the unique solution of the problem: find $u \in V$ such that

$$J(u) = \min_{v \in V} J(v) \tag{1.27}$$

where $J(v)$ is the quadratic functional defined by:

$$J(v) = \frac{1}{2} a(v,v) - L(v)$$

The most important application where this happens is in linear elasticity, in which case the variation equation (1.4) represent the *principle of virtual work* and the functional J represents the *total energy* of the body. This example is important also because "its variational formulation is basically responsible for the invention of the finite element method by engineers" [41].

The Galerkin method is often applied also outside linear elasticity problems but, as we will see in the following Section, when the optimality condition (1.25) cannot be satisfied (e.g. when the bilinear form is not symmetric) the approximate solution u_h can result to be completely unsatisfactory.

1.3 (Galerkin) Faults and Remedies

The Galerkin method is optimal for some applications but is unsatisfactory for others, as reported in the literature ([25] and [97] Chapter 2). Here we take in consideration the case of the linear, steady advection-diffusion equation, which brings to second order partial differential boundary-value problems which do not belong to the class (1.7). In particular the resulting bilinear form is non-symmetric (while for the class of problems (1.7) the corresponding bilinear form is symmetric if and only if $a_{ij} = a_{ji}$). Let us consider the following boundary value problem in one-dimension:

$$\nu \frac{\partial u}{\partial x} = k \frac{\partial^2 u}{\partial x^2} + f \tag{1.28}$$

where ν is the advection coefficient, k is the diffusivity coefficient and the Dirichlet boundary conditions $u(0) = g_0 \quad u(L) = g_L$ are imposed.

The exact solution $u(x)$ when the source term applied is null, i.e. ($f = 0$), is

$$u(x) = c_1 + c_2 e^{P \frac{x}{L}} \tag{1.29}$$

where c_1 and c_2 are constants and $P = \nu L/k$ is the *Peclet number*. It indicates the balancing between advection and diffusion, i.e. when $P > 1$ the flow is advection-dominated, while when $P < 1$ the flow is diffusion dominated. The variational formulation for this problem, corresponding to (1.14), becomes: find $u_h \in S_h$ such that for all $w_h \in V_h$ it holds

$$\int_0^L \left(-\frac{\partial w_h}{\partial x} \nu u_h + \frac{\partial w_h}{\partial x} k \frac{\partial u_h}{\partial x} \right) dx = \int_0^L w_h f dx \tag{1.30}$$

which, after another integration by parts, becomes:

$$\sum_{A=1}^{A_{max}} \int_{x_{A-1}}^{x_A} w_h (\nu \frac{\partial u_h}{\partial x} - k \frac{\partial^2 u_h}{\partial x^2} - f) dx - \sum_{A=1}^{A_{max}-1} w_h(x_A)(k \frac{\partial u_h}{\partial x}(x_A^+) - k \frac{\partial u_h}{\partial x}(x_A^-)) = 0 \tag{1.31}$$

where x_A are the nodal-points of the discretization. This bilinear form is *V-elliptic* since

$\forall w \in V_h$:

$$\begin{aligned} a(w,w) &= \int_0^L \left(-\frac{\partial w}{\partial x}\nu w + k(\frac{\partial w}{\partial x})^2 \right) \\ &= \int_0^L \left(-\nu/2 \frac{\partial (w^2)}{\partial x} + k(\frac{\partial w}{\partial x})^2 \right) \\ &= -\tfrac{\nu}{2}(w^2)|_0^L + k||\frac{\partial w}{\partial x}||^2 \\ &= k||\frac{\partial w}{\partial x}||^2 \\ &\geq k/C||w||^2 \end{aligned} \quad (1.32)$$

where the last inequality is obtained applying the Poincare'-Friedrichs inequality. Then, if $k > 0$ the problem still have a unique solution u_h, i.e. the Lax-Milgram Lemma still holds.

But the quality of the approximation u_h when the Galerkin method is used, is no longer guaranteed. In fact, the bounds given by the Cea's Lemma are practically useless in the advection dominated case, since the values of the constants α and M make the right-hand side very large [25].

Furthermore, since the bilinear form $a(\cdot,\cdot)$ is not symmetric anymore, the condition (1.24) does not correspond to an orthogonality condition and the optimality condition (1.25) is not necessarily satisfied. It turns out in practice that solving this problem applying the standard Galerkin Finite Element formulation works for the diffusion-dominated case but in the advection-dominated case it shows a wildly oscillatory response (see Figure 1).

To avoid this problem, some authors (see e.g. [27]) suggest to refine properly the discretization in such a way that an element Peclet number condition on the mesh size is satisfied or at least in regions where are present boundary or interior layers in which the solution and its derivatives are changing significantly. An uniform refinement to satisfy the element Peclet-number condition would be too expensive, while an adaptive refinement would be better, but anyway this solution seems not to be optimal, because for example the oscillations could lead to unnecessary refinement outside the layers.

A successful Finite Element method to solve problem (1.30) has been derived in [72] and brought to the so-called *stabilized methods (SUPG/GLS)* (see [25] and references therein). They consist on choosing a proper space of weighting functions W_h, not necessarily concident with the space V_h in which it is contained the approximate solution u_h (which is the choice of the Galerkin method). The choice of the weighting-functions space has been carried out following considerations about the physics of the problem, which bring to the formulation of upwind weighting-functions. An important property of SUPG is that by only modifying the weighting functions space it remains a weighted residuals method applied to the underlying differential problem, i.e. the *consistency* is not lost.

The SUPG method brings to the following weak formulation of the problem: find $u_h \in V_h$

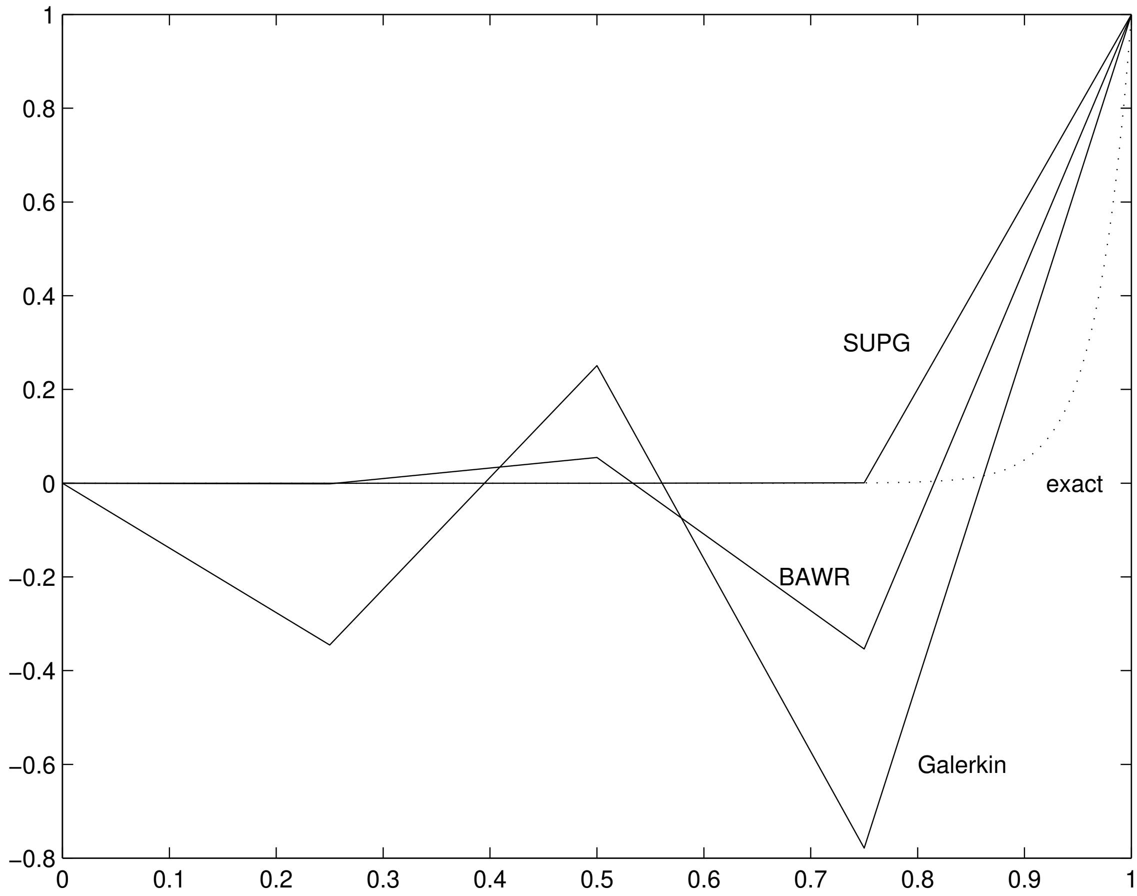

Figure 1: various finite-element discretization methods applied to problem (1.28) in the advection-dominated case

such that for all $w_h \in W_h$ it holds

$$\int_0^L \left(-\frac{\partial w_h}{\partial x} \nu u_h + \frac{\partial w_h}{\partial x} k \frac{\partial u_h}{\partial x} \right) dx + \sum_{A=1}^{A_{max}} \int_{x_{A-1}}^{x_A} p_h (\nu \frac{\partial u_h}{\partial x} - k \frac{\partial^2 u_h}{\partial x^2} - f) dx = \int_0^L w_h f \, dx \tag{1.33}$$

i.e. a term is added to the formulation (1.30), which after integration by parts becomes:

$$\sum_{A=1}^{A_{max}} \int_{x_{A-1}}^{x_A} (w_h + p_h)(\nu \frac{\partial u_h}{\partial x} - k \frac{\partial u_h}{\partial x} - f) dx - \sum_{A=1}^{A_{max}-1} w_h(x_A)(k \frac{\partial u_h}{\partial x}(x_A^+) - k \frac{\partial u_h}{\partial x}(x_A^-)) = 0 \tag{1.34}$$

and it shows explicitly, if compared with (1.31), that the method correspond to a modification of the weighting functions (for element interiors only).

The p_h are taken as $p_h = \tau \nu \frac{\partial w_h}{\partial x}$, where τ (called the *intrinsic time-scale* since it has the dimension of time) can be chosen e.g. to obtain the exact artificial diffusion (i.e. the correspondence with the exact solution when $f = 0$ on a uniform mesh):

$$\tau = \frac{\tilde{k}}{|\nu|^2} \tag{1.35}$$

Figures 3 and 4 show the effect of the additional term on the weighting functions used in the SUPG method. It is important to note that, in the advection-dominated case,

$$\int_{x_A}^{x_{A+1}} N_A^{SUPG} dx = 0 \qquad \text{and} \qquad \int_{x_{A-1}}^{x_A} N_A^{SUPG} dx = h \tag{1.36}$$

that is, the weight is confined to the element upwind of node A. It can be demonstrated [60] that the bilinear form which derives from the SUPG method have a better stability than that which derives from the Galerkin method (note that the *V-ellipticity* condition (1.2) is called in this case also *stability* condition). We have:

$$a_{SUPG}(w, w) \geq \frac{1}{2}(k + \tau\nu^2)||\frac{\partial w}{\partial x}||^2 \tag{1.37}$$

that must be compared with (1.32). The SUPG method has been generalized to be applied to a broader class of differential operators (e.g. compressible and incompressible Navier-Stokes equations), see e.g. references contained in [25]. Another noteworthy approach to solve problems for which the Galerkin method fails is to enrich the space of functions used in the standard Galerkin method, e.g. with *residual-free bubble functions* [25]. We do not consider these methods here because they are outside the scope of this Chapter and their effectiveness is shown to be equivalent to that of the *stabilized methods* previously presented (see [26] and references therein).

In the following Section it is presented an analytic framework to obtain optimal solutions even when the bilinear form $a(\cdot, \cdot)$ is unsymmetric.

1.4 The Best-Approximation Weighted-Residuals (BAWR) Method

We have seen in Section 1.1 that, starting from the weighted-residuals formulation (1.23) and applying the Green's Lemma, we arrive to the variational formulation (1.4). Then, if the bilinear form $a(\cdot, \cdot)$ contained in the variational formulation is symmetric, the Galerkin method finds the optimal solution which arises naturally from this formulation, i.e. in the norm induced by the inner-product corresponding to the bilinear form $a(\cdot, \cdot)$ contained in the variational formulation.

In this Section it is considered the problem of finding the optimal solution that arises naturally from the weighted-residuals formulation(1.23), hence before applying the Green's Lemma. To derive the best-approximation (i.e. the orthogonality of the error with respect to the approximating subspace) with respect to this formulation, we follow the approach of the *stabilized methods*, i.e. we choose an appropriate weighting-functions space. In practice here it is presented an analytic method to derive the weighting functions that bring to the best-approximation of the true solution in the norm induced by the inner-product adopted in the weighted-residuals formulation. It must be noted that here the optimization is always possible, since an inner-product is formed to weight the residuals by definition, while the bilinear form defines an inner-product only if it is symmetric.

Usually, the standard weighted-residuals formulation adopted in the Finite Element method uses the L_2 inner-product, hence in the following we will refer directly to the optimization in the L_2-norm. The main result is contained in Theorem 1.1 .

Let us consider the following model problem (for simplicity in only one dimension): find $u \in V$ such that

$$\begin{aligned} \mathcal{L}u = f \qquad & on \quad \Omega \\ u = u_g \qquad & on \quad \Gamma_g \\ \tfrac{du}{dx} = u_n \qquad & on \quad \Gamma_n \end{aligned} \tag{1.38}$$

where $\Omega \subset R$, $\mathcal{L}$ is a one-to-one [95] linear partial differential operator with its adjoint $\mathcal{L}^*$ and V is an Hilbert space of functions like in Section 1.1. A weighted-residuals formulation of this problem, becomes: find $u \in V$ such that

$$(w, \mathcal{L}u - f) = 0 \qquad \forall w \in W \tag{1.39}$$

where W is also an Hilbert space of weighting-functions, $(\cdot,\cdot)$ is the inner product in L_2 and the same boundary conditions are applied. As before, to numerically approximate this problem two finite-dimensional subspaces $V_h \subset V$ and $W_h \subset W$ are chosen and the following problem is solved: find $u_h \in V_h$ such that

$$(w_h, \mathcal{L}u_h - f) = 0 \qquad \forall w_h \in W_h \tag{1.40}$$

The following Theorem tells how to choose the weighting-functions space W_h in such a way that the approximation error $u - u_h$ is orthogonal to the space V_h of the approximating functions.

THEOREM 1.1: " *Given the model problem (1.38), if we solve it numerically using the weighted-residuals formulation (1.40) and we choose the weighting functions $V_h = \{w_h\}$ as the solutions of the boundary-value-problems (the adjoint problems): for each $\phi_h \in V_h$, find $w_h \in W_h$ such that*

$$\mathcal{L}^* w_h = \phi_h \tag{1.41}$$

where $\mathcal{L}^*$ is the adjoint of $\mathcal{L}$ and the adjoint boundary conditions are imposed, we obtain that

$$(\phi_h, u - u_h) = 0 \qquad \forall \phi_h \in V_h \qquad . \tag{1.42}$$

Moreover, the $\{w_h\}$ form a basis for W_h. "

Proof: let us assume for now that the boundary conditions be homogeneous; the general case will be demonstrated to be equivalent at the end of the demonstration. With homogeneous boundary conditions, by definition of adjoint problem [95], it holds:

$$(\mathcal{L}^* w_h, u_h) = (w_h, \mathcal{L} u_h) \qquad \forall u_h \in V_h \tag{1.43}$$

if adequate *adjoint boundary conditions* are imposed on w_h.

Recalling the expression which characterizes the weighted-residuals formulation (1.23), we have

$$0 = (w_h, \mathcal{L}(u - u_h)) = (\mathcal{L}^* w_h, u - u_h) \tag{1.44}$$

Now, given a base $\{\phi_h\}$ for V_h, the finite-dimensional space of approximating functions, we solve the *adjoint problems*: for each $\phi_h \in S_h$, find $w_h \in V_h$ such that

$$\mathcal{L}^* w_h = \phi_h \tag{1.45}$$

and imposing the *adjoint boundary conditions* which make true the second equality in (1.44) [95].

If the $\{w_h\}$ are used as the weighting-functions in the weighted-residuals problem (1.40), from (1.44) we obtain the final result

$$(\phi_h, u - u_h) = 0 \qquad \forall \phi_h \in S_h \qquad . \tag{1.46}$$

Then, since the $\{\phi_h\}$ are linearly independent and $\mathcal{L}^*$ is one-to-one by hypothesis, also the $\{w_h\}$ are linear independent. Moreover, since the dimensions of V_h and of W_h are the same, the $\{w_h\}$ form a base for W_h.

Now let us consider the case of non-homogeneous boundary conditions as in the model problem (1.38). From the general weak formulation (1.14) of a boundary value problem of the class here considered, we see that non-homogenenous boundary conditions modify the right-hand-side of the formulation, i.e. they correspond to an homogeneous problem with a different forcing term applied.

Since the result of this theorem holds for whatever choice of the right-hand-side (as can be easily verified from (1.14)), it holds also for whatever choice of the Dirichlet and Neumann boundary conditions. $\diamondsuit$

The above Theorem holds in general for weighted-residuals methods. Here we will consider the case of V_h being a space of finite elements. Then the approximation method considered is the Finite Element method.

The result of Theorem 1.1 tells us that the approximation error is orthogonal to the space of approximating functions V_h with respect to the L_2-norm. Since V_h is an Hilbert space, with this property the well known *Projection Theorem* guarantees that u_h is the best approximation [47], or equivalently, the optimal approximation [86], of u in the L_2-norm, and this best-approximation always exist, in the hypotheses made, and is unique.

Let us show, as an example, the solution, using a finite element approximation in which the weighting-functions are chosen according to theorem 1.1, of the advection-diffusion problem (1.28). The *adjoint problems* here become: for each $\phi_h \in V_h$, find $w_h \in W_h$ such that

$$\begin{aligned} -k\frac{\partial^2 w_h}{\partial x^2} - \nu\frac{\partial w_h}{\partial x} &= \phi_h \\ w_h(0) = w_h(L) &= 0 \end{aligned} \tag{1.47}$$

These *adjoint problems* are boundary-value problems defined on the same domain of the original problem, but with a very simple (usually a low-order polynomial) and highly localized forcing term. Their numerical solution can be done e.g. using a finite elements approximation, i.e. find $w_h \in V^b{}_h$ such that for all $\chi_h \in W^b{}_h$ (e.g. a piecewise linear weighting-functions space) we have

$$\int_0^L \left(-\frac{\partial \chi_h}{\partial x}\nu w_h + \frac{\partial \chi_h}{\partial x}k\frac{\partial w_h}{\partial x} \right) dx = \int_0^L \chi_h \phi_h dx \tag{1.48}$$

Now, if $V^b{}_h$ is a piecewise-linear space of functions and to solve (1.48) it is used the same discretization adopted for the original problem (1.28), i.e. $V^b{}_h = V_h$, the approximation of the right-hand-side $\phi_h \in V_h$ it is made with functions $L^* w_h$ which are piecewise-constant functions (plus distributional terms). If the discretization used for V_h is not sufficiently fine, also the resulting weighting-functions are not a good approximation of the *adjoint problems* and the result does not hold.

To avoid this problem two are the possibilities. The first is to adopt a finer discretization than that used for V_h, while the second is to use higher order polynomials on the same discretization. It is important to note that, in either cases, when the weighting-functions will be used to solve the original problem, in the discretization defined for V_h they will appear as non-linear functions.

From the computational point-of-view, the strong locality of the right-hand-side in the adjoint problems let us think that also the solution will be only locally significative, thus requiring a reduced computational cost (cfr. Part III of this Thesis). Furthermore, the experimental evidence tells that the weighting-functions (i.e. the solutions of the *adjoint problems*) can be very roughly approximated (even zeroed) outside the subdomain of definition of the ϕ_h. Hence it turns out that the problems (1.47), which at a first sight seem to require a computational cost analogous to that of the overall original problem, have actually a cost comparable to that of very local problems (i.e. few equations each to be solved).

Moreover, the *adjoint problems* are very similar among them: the forcing term is very local and always the same; it only changes its "point-of-application". Hence the solutions w_h can be e.g. quite easily classified with respect to the distance from the boundary of the corresponding ϕ_h and the actual computation can be performed only for one member of each class. It is important to note that this means also that the weighting-functions chosen according to Theorem 1.1 adapt themselves to the geometry of the domain. The

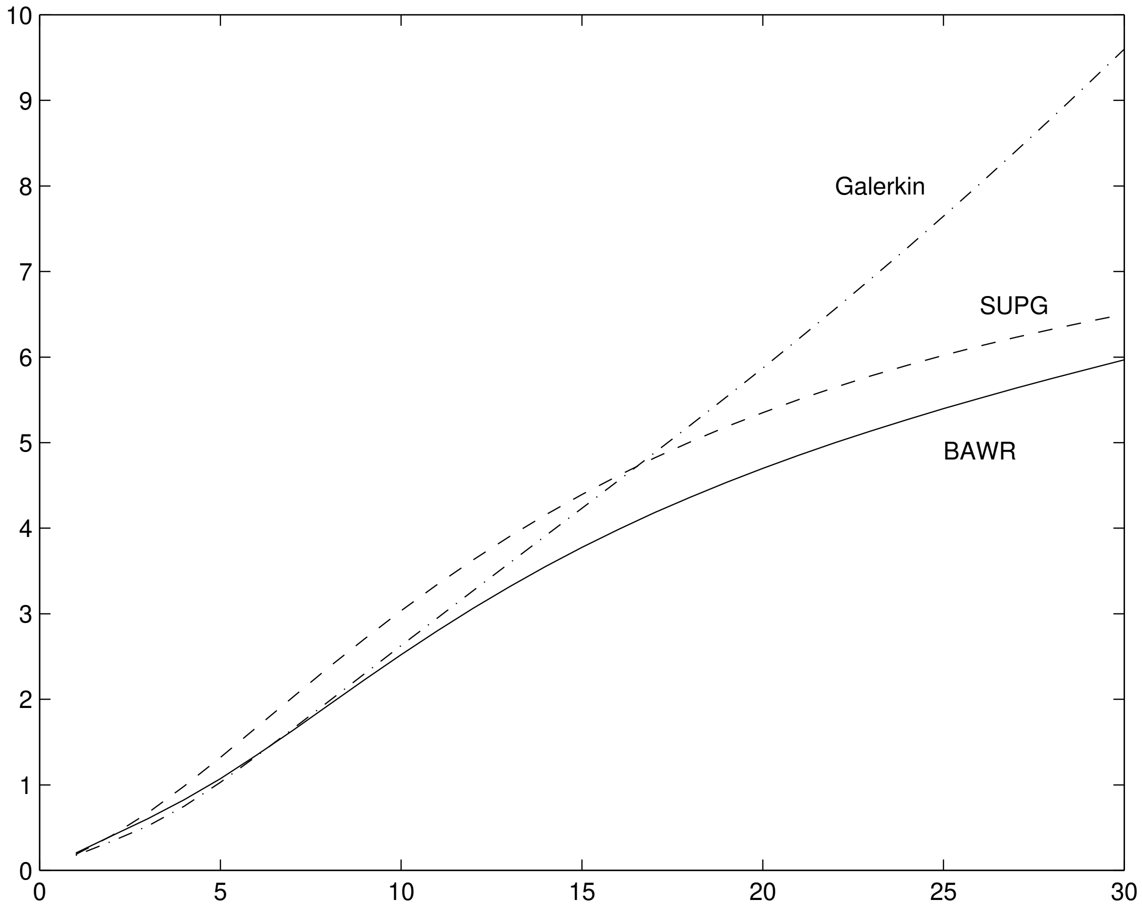

Figure 2: the global approximation error in the L_2-norm for the problem (1.28) with different Peclet numbers

importance in applications of this aspect will be investigated in the future, when 2-D and 3-D problems will be addressed.

In Figure 2 are presented some experimental results obtained for the problem (1.28): the global error $||u - u_h||_{L^2}$ obtained with three finite element methods and different Peclet numbers (on X-axis). With “BAWR” it is indicated the finite element method corresponding to the choice for the weighting-functions dictated by Thorem 1.1. The same curve (i.e. with almost exact superposition) has been obtained calculating the optimal w_h analytically or numerically using piecewise-quadratic finite elements. In this case the truncation of w_h outside the support of the corresponding ϕ_h has given also the same result. This is computationally very important, since in this case the system matrix of the original problem remains sparse as in the usual finite element methods.

1.5 Conclusions and further Developments

In this Chapter it has been discussed the influence of the choice of the weighting-functions space on the optimality of the approximation error obtained in a finite-elements discretization of a boundary-value problem. This is not a topic frequently discussed in the standard finite-elements literature, since usually it is studied the choice of the finite elements space (i.e. the approximating functions space) to use for achieving the desired approximation result, and the the weighting-functions space adopted is by default that of the Galerkin method. Actually, the choice of the finite elements discretization determines the approximation power adopted to solve the problem, while the choice of the weighting-functions space determines how this "power" is used to get in practice the numerical solution. Hence, the optimality criterion of the approximation is decided by the choice of the weighting-functions space. With this respect, the method that has been presented in this Chapter is an optimal method alternative to the Galerkin method e.g. for all the operators which lead to an unsymmetric bilinear form in the corresponding weak formulation of the differential problem.

The analytic method presented in this Chapter permits also a more general analysis of weighted-residuals methods. For example, within the framework presented in Theorem 1.1 it is possible to give an interpretation of the Galerkin and SUPG methods, as we will see below. Let us start with the Galerkin method:

Lemma 1.2: *"Let $V^p{}_h$ be the space of piecewise-linear functions defined on the same discretization used for V_h (which is also a space of piecewise-linears functions). This corresponds to the choice of the weighting-functions in the Galerkin method. In this case, the method for choosing the weighting functions $w_h \in V_h$ described in Theorem 1.1 gives the weighting functions used in the Galerkin method.*

Moreover, for the Laplace operator the Galerkin method coincides with the method presented in Theorem 1.1 applied to the problem governing the solution derivatives. "

Proof: the result of Theorem 1.1 is obviously independent from the basis adopted for the subspace generated by the $\{w_h\}$. If the weighting functions given by this method are chosen within piecewise linears in the same discretization used for S_h, the weighting functions of the Galerkin method (which are either piecewise linear and with the same cardinality and support) form also a basis for V_h and therefore the results obtained using the two basis of weighting functions are coincident.

In the case of the Laplacian, the problem governing the solution derivatives becomes:

$$-\frac{\partial \nabla u_h}{\partial x} = f \tag{1.49}$$

and the *adjoint problems* becomes:

$$\frac{\partial \nabla w_h}{\partial x} = \nabla \phi_h \tag{1.50}$$

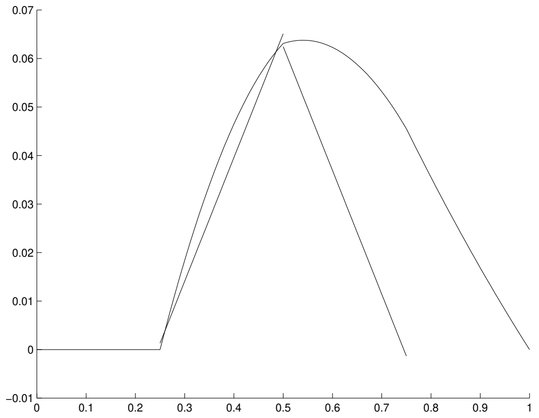

Figure 3: SUPG (linears) and BAWR weighting functions in the diffusion-dominated case, for a node centered at $x = 0.5$

and the application of these weighting-functions to the solution of the Laplacian, gives:

$$(\nabla\phi_h, \nabla u_h - \nabla u) = 0 \tag{1.51}$$

which coincides with the Galerkin ortogonality condition (1.24):

$$a(\phi_h, u_h - u) = 0 \tag{1.52}$$

$\diamondsuit$

The coincidence of the two methods is also verified experimentally.

Note that in the case considered in Lemma 1.2, the *adjoint problems* may be not well approximated, since the residual of the strong form is the difference between two polynomials of different degree (recall the analogous consideration which follows the demonstration of Theorem 1.1). It is to be noted the great importance of the fact that the *adjoint problems* of Theorem 1.1 must be "well approximated". If the Galerkin method is used, this implies that the discretization must be sufficiently fine. Note that analogous considerations were made in Section 1.3, before presenting the SUPG method.

The interpretation of the SUPG method within the framework of Theorem 1.1 is a little bit more complicated. Let us start by the observation that SUPG uses discontinuous weighting-functions, i.e. it corresponds to a non-conforming finite elements approximation [73] of the *adjoint problems*. These weighting-functions are built by adding a constant term p_h (cfr. Section 1.3) to the Galerkin piecewise linears weighting-functions. This additional "global degree-of-freedom" is the key of the success of the SUPG/GLS methods. Here we argue that its choice results in a linear approximation (e.g. in the mean) of the weighting functions that arise from the application of Theorem 1.1 .

Let us consider again the advection-diffusion equation (1.28) with homogeneous Dirichlet boundary conditions and the resulting adjoint problems (1.47). Let us assume that the

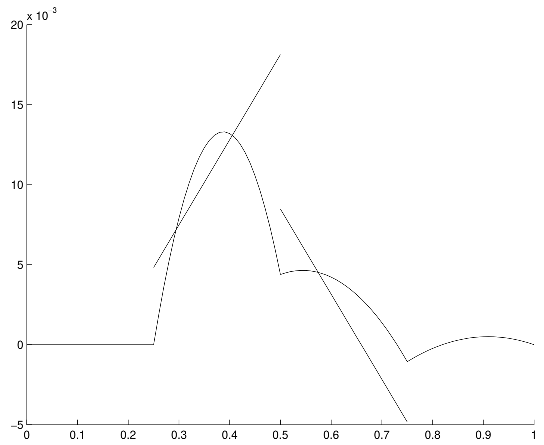

Figure 4: SUPG (linears) and BAWR weighting functions in the advection-dominated case, for a node centered at $x = 0.5$

weighting-functions used for the BAWR method are taken from the piecewise quadratic functions space, defined on the discretization in which is defined V_h. On each subdomain in which the ϕ_h are defined, they must satisfy the equation of the adjoint problem (1.47) and then we choose the function that vanish on the boundary of the inteval (in this way we have three conditions which defines univocally the second order polynomial). Let us calculate the integral of the BAWR weighting-function on the subdomain. Next we determine the value of p_h that gives the equivalence in the mean with the BAWR weighting-functions: when the advection highly dominates, we have $p_h = 1/2$ which is the same value of SUPG; in the other cases, the numerical comparisons do not agree satisfactorily. Further investigations are necessary. In Figg. 3 and 4 a visual comparison between the two weighting-functions is, however, indicative.

Further extensions of the BAWR method could go in many directions (e.g. following the path which have done the stabilized methods during the last fiftheen years, mainly through fluid dynamics problems). Here we make only the further consideration that a straightforward extension of the method to non-linear problems is possible, making use e.g. of the Frechet derivative of the non-linear operator [95].

Chapter 2

2

A-posteriori Error Estimation

In this Chapter we will address the problem of estimating the approximation error that arises when the solution $u \in S$ of an exact variational approximation problem is approximated by a function $u_h \in S_h$ (see Section 1.1 for the details) and S_h is a finite elements space.

The estimates of the approximation error $u - u_h$ are necessarily derived in a specified norm. Common choices are the *energy*-norm

$$||u - u_h||_E = (a(u - u_h, u - u_h))^{1/2} \qquad \forall u_h \in V_h \tag{2.1}$$

and the L_2-norm.

$$||u - u_h||_2 = (\int_\Omega (u - u_h)^2 d\Omega)^{1/2} \tag{2.2}$$

The error estimates here presented have two fundamental tasks in every adaptive algorithm (whose general schema has been presented in the Introduction). First, they drive the local adaptation of the discretization, i.e. it is their responsibility to *indicate* where it is necessary/convenient to add/remove degrees-of-freedom from the discretization. Second, they must assure that the final dicretization approximate (in the selected norm) the true solution within the desired tolerance, i.e. they must provide sharp and robust *estimates* of the global approximation error. It is common to consider the contribution to the approximation error as a quantity related to each finite-element. From the techniques presented in this Chapter are therefore derived the *error indicator* η_K, relative to element K, for the first task, and the *error estimator*

$$\eta^2 = \sum_{K=1}^{N_el} {\eta_K}^2 \tag{2.3}$$

for the second task.

The quality of an error estimator is measured by the *effectivity index*

$$\theta = \frac{\eta}{||u - u_h||} \tag{2.4}$$

When $\theta \to 1$ as $||u - u_h|| \to 0$ then the stimator is said to be *asymptotically exact*.

While in the past the attention of the researchers was directed to the discovery of new methods for a-posteriori error estimation, now there is more attention to validate the robustness of the estimators (theoretically and experimentally).

The computational cost of the estimator is a main constraint for its success, since it is an auxiliary operation to the computation of the solution and then it must not increase significantly the total cost.

As will be shown in the sequel, every estimator requires to compute *extensions* [7] of the computed solution and this comes from the solution of a large linear system of equations. The solution of this system represents a considerable percentage of the total cost required by the finite element method. For this reason, in this Chapter it is investigated the possibility to use the solver dedicated to adaptive updates (which correspond to *local extensions*) presented in the Part III of this Thesis. It turns out the possibility to define an a-posteriori error estimator with more potentials than the existing ones.

2.1 Background

Every adaptive algorithm needs to estimate the approximation error to do its job correctly, i.e. adapt the finite element mesh to get a solution of prescribed accuracy with the minimum number of *degrees of freedom* necessary.

2.1.1 *Extensions*

A key issue in both error control and estimation is that of *extension* of the finite element mesh [7]. Recall that a *finite element* is characterized by its geometric shape plus the approximating functions (called *shape functions*) defined on it. Therefore a possible extension performed by an adaptive algorithm may correspond to increase locally the density of elements through a mesh-refinement operation (*h-extension*), or to increase the number of shape functions defined for a group of adjacent elements (*p-extension*) or a combination of both (*hp-extension*). In the following we will take into consideration only h-extensions, unless otherwise stated. In general, after an extension is made, the resulting finite element space has improved approximation properties.

Note that the term “locally” is not embedded in the concept of extension, which can be done e.g. uniformly in the overall domain. In this case it is called a *uniform* extension. A uniform extension is used to measure experimentally the rate of convergence of an algorithm on a specific problem or class of problems. A *local* extension is generally used by adaptive algorithms.

Performance of extensions, considered e.g. in term of the relationship between the error measured in some norm (energy, L_2, L_∞,...) and the number of degrees of freedom N, is of paramount importance for the estimation of computing time and resources needs.

2.1.2 A-Priori Analysis

Let us see in a simple example an a-priori analysis of the approximation error for uniform h-extensions [7]. Given the one-dimensional boundary-value problem:

$$-a \cdot \frac{\partial^2 u}{\partial x^2} + cu = f \quad x \in [0, L]$$

with appropriate boundary conditions. Let us assume that a, c and f are such that $\left|\dfrac{\partial^2 u}{\partial x^2}\right| \leq C$.

The weak formulation for this problem (cfr. Chapter 1), becomes:

$$\int_0^L (a\frac{\partial u}{\partial x}\frac{\partial v}{\partial x} + cuv)dx = \int_0^L fvdx \qquad \forall v \in V \tag{2.5}$$

Consider an uniform subdivision of the interval $[0, L]$. The length of the element is therefore $h = L/n$, where n is the number of elements. Be u_n the piecewise linear interpolant of u, i.e.

$$u_n(jh) = u(jh) \qquad j = 0, 1, \ldots, n$$

and the interpolation error on the k-th element

$$\bar{e_k} = u(x) - u_n(x) \qquad (k-1)h \leq x \leq kh \quad k = 1, 2, \ldots, n$$

Since $\bar{e}_k(x)$ vanishes at the endpoints of the element , there is a point $\bar{x}_k$ where $|\bar{e}_k|$ is maximal. In this point $\bar{e_k}' = 0$. Then:

$$\bar{e}_k^{'}(x) = \int_{\bar{x}_k}^{x} \bar{e}_k^{''}(t)dt = \int_{\bar{x}_k}^{x} \bar{u}^{''}(t)dt \qquad (k-1)h \leq x \leq kh \tag{2.6}$$

Since $|u^{''}| \leq C$, we have

$$max|\bar{e}_k^{'}(x)| \leq hC \qquad (k-1)h \leq x \leq kh$$

Let us now expand $\bar{e}_k$ into a Taylor series about the point $\bar{x}_k$, and let us assume that $\bar{x}_k$ is located such that $kh/\bar{x}_k \leq h/2$. We have:

$$\bar{e}_k(kh) = 0 = \bar{e}_k(\bar{x}_k) + (kh - \bar{x}_k)\bar{e}_k^{'}(\bar{x}_k) + \frac{1}{2}(kh - \bar{x}_k)^2\bar{e}_k^{''}(t)$$

where t is a point between $\bar{x}_k$ and $x = kh$. Because $\bar{e}_k^{'}(\bar{x}_k) = 0$:

$$max|\bar{e}_k(\bar{x}_k) = \frac{1}{2}|kh - \bar{x}_k|^2|\bar{e}_k^{''}(t)| \leq \frac{h^2}{8}C \tag{2.7}$$

If $\bar{x}_k$ is closer to $(k-1)h$ than to kh, then we write the Taylor series expression for $\bar{e}_k((k-1)h)$ instead of $\bar{e}_k(kh)$ and obtain the same result. Equation (2.7) is a basic result of interpolation theory. In view of (2.6) and (2.7), the strain energy of the error of $\bar{e}$ is:

$$E(\bar{e}) = \frac{1}{2}\int_0^L (a\cdot(\bar{e}')^2 + c\bar{e}^2)dx = \frac{1}{2}\sum_{k=1}^{n}\int_{(k-1)h}^{kh}(a\cdot(\bar{e}_k')^2 + c\bar{e}_k^2)dx \leq \frac{1}{2}nh\left(K_1(Ch)^2 + K_2\left(\frac{h^2}{8}C\right)^2\right)$$

where K_1 and K_2 are constants chosen so that $a(x) \leq K_1$ and $c(x) \leq K_2$ in the interval $[0, L]$. Clearly, K_1 and K_2 are independent of h. Since $nh = L$, there is a constant K such that:

$$E(\bar{e}) \leq \frac{1}{2}KLC^2h^2$$

Finally, since the energy norm of the error of the finite element solution (2.1), using the Galerkin method with the kind of problem here considered, is less than or equal to the energy norm of the error of $\bar{e}$ (cfr. (1.25)), we have:

$$||e||_E = \sqrt{E(e)} \leq kCh$$

where C depends on a, c and f but is independent of h. This is the typical a-priori estimate. They are based on some general information about the exact solution. In this case it was $\left|\frac{\partial^2 u}{\partial x^2}\right| \leq C$. In real problems C is not known or known very inaccurately. A priori estimates are useful to indicate how fast the error changes as the discretization has been changed. Since the mesh is uniformly refined it holds $N \approx L/h$ (in 1D) and it is possible to express the majorization of the error in terms of the number of degrees of freedom:

$$||e||_E \leq \frac{k}{N} \tag{2.8}$$

2.1.3 Convergence Rates

This topic is introduced with an example [7]. Let us consider a class of problems, taken from (2.5), for which the exact solution is:

$$u = \begin{cases} -c\frac{x}{L}\left(1-\frac{x_a}{L}\right)^\lambda & x \leq x_a \\ c\left(\frac{x}{L}-\frac{x_a}{L}\right)^\lambda - c\frac{x}{L}\left(1-\frac{x_a}{L}\right)^\lambda & x > x_a \end{cases} \tag{2.9}$$

where c is a constant and $0 \leq x_a \leq L$. If $\lambda \geq 2$ the solution is *smooth* and the a-priori analysis is valid. If $1/2 < \lambda < 2$ the solution is *non-smooth*, but $E(u) < \infty$ and the corresponding solutions are admissible for the finite element approximation. In this case, the relation (2.8), becomes:

$$||e||_E \leq \frac{k}{N^\beta} \tag{2.10}$$

where k and β are constants independent of h, and:

$$\beta = min\left(p, \lambda - \frac{1}{2}\right) \tag{2.11}$$

where p is the maximum degree of the polynomials used within the element (here $p = 1$). The form of expression (2.10) is typical of h- and p-extensions. These estimates becomes sharp when N is large, i.e. we can substitute the $\leq$ sign with the $\approx$ sign. Taking the logarithms of both sides of (2.10), we have:

$$log||e||_E \approx logk - \beta logN \tag{2.12}$$

The absolute value of the slope, β is called the *asymptotic rate of convergence* in the norm utilized. The rate of convergence is a measure of how difficult it is to control the error (in this example, in the energy norm).

From the expression (2.11) we see that when the solution is *smooth* the rate-of-convergence is determined by the maximum degree of the polynomial used within the element, while when the solution is *non-smooth* the rate is lower.

This lower rate-of-convergence in the case of non-smooth solution may be increased to its optimal value by adaptive mesh refinement:

$$||e||_E \leq \frac{k}{N^p} \tag{2.13}$$

which, in fact, depends again only on the maximum degree of the polynomials used inside the elements.

It must be noted that to quantify the cost of an adaptive analysis it is not sufficient to consider the final number of degrees-of-freedom and to multiply it by an average cost-per-d.o.f.; this is because it must be considered also the intermediate steps which have been done to reach the final mesh. These intermediate steps are determined by the *adaptive algorithm*, which must solve in substance an optimization problem which is in general too complex to formalize mathematically. Hence some heuristics are unavoidable in the algorithm formulation.

In general, however, the less smooth the exact solution, the greater the benefits gained through the use of adaptive methods.

2.2 Overview of A-Posteriori Error Indicators and Estimators

To present some approaches that bring to an a-posteriori estimation of the approximation error we will take in consideration its estimation in the *energy*-norm (2.1).

The presentation of the estimators will be made considering the following model problem:

$$-\Delta u = f \quad in \quad \Omega$$
$$u = 0 \quad on \quad \Gamma_D$$
$$\frac{\partial u}{\partial n} = g \quad on \quad \Gamma_N$$

in a connected, bounded, polygonal domain $\Omega \subset R^2$ with boundary $\Gamma = \Gamma_D \cup \Gamma_N, \Gamma_D \cap \Gamma_N = 0$. Let it be

$$V = \{\phi \in H^1(\Omega) : \phi = 0 \quad on \quad \Gamma_D\}$$

then the Galerkin method (a natural choice for this type of model problem) brings to the following weak form of the problem (cfr (1.14)): find $u \in V$ such that

$$\int_\Omega \nabla u \cdot \nabla v = \int_\Omega f v + \int_{\Gamma_N} g v \qquad \forall v \in V \tag{2.14}$$

Denote by V_h the space of all continuous, piecewise linear functions corresponding to a *regular family* [41] of finite elements, vanishing on Γ_D. We then obtain the resulting approximate variational problem: find $u_h \in V_h$ such that

$$\int_\Omega \nabla u_h \cdot \nabla v_h = \int_\Omega f v_h + \int_{\Gamma_N} g v_h \qquad \forall v_h \in V_h \tag{2.15}$$

Let $u \in V$ and $u_h \in V_h$ be the exact solutions of Problems (2.14) ad (2.15), respectively. They satisfy the identity:

$$\int_\Omega \nabla(u - u_h) \cdot \nabla v = \int_\Omega f v + \int_{\Gamma_N} g v - \int_\Omega \nabla u_h \cdot \nabla v \qquad \forall v \in V \tag{2.16}$$

which in compact form, becomes:

$$a(e, v) = L(v) - a(u_h, v) \qquad \forall v \in V \tag{2.17}$$

Here the right-hand side implicitly defines the residual of u_h as an element of V' (i.e. the dual space of V). It is easy to demonstrate [100] that the norm in V of the error is, up to multiplicative constants, bounded from above and from below by the norm of the residual in V' (cfr (1.3)).

The relevant thing is that the identity (2.17) is a *model* which describes the error in V using only known or computed data (i.e. the *residual*). Most a-posteriori error estimators try to estimate this dual norm of the residual by quantities that can more easily be computed from f, g, and u_h.

It is important to recall that, since $V_h \subset V$ and $a(\cdot, \cdot)$ is symmetric and the Galerkin method has been used, then the error is orthogonal to V_h (cfr. (1.24)):

$$a(e, v_h) = \int_\Omega \nabla(u - u_h) \cdot \nabla v_h = 0 \qquad \forall v_h \in V_h \tag{2.18}$$

hence the residual vanishes on V_h.

In the following there is a presentation of some a-posteriori error estimators. This is not intended to be exhaustive, but it covers the main ideas. Further material can be found in the cited literature. It can be noted that often the error estimates are derived in the energy norm. For error estimation in the L_2-norm, these estimates are then "converted" using a duality technique which is due to Aubin and Nitsche (see e.g. [96] [2]). If the BAWR method is used (cfr. Chapter 1), the orthogonality property given in Theorem 1.1 can let us to derive error estimates directly in the L_2-norm. Moreover, this holds for problems which lead to both symmetric and un-symmetric bilinear forms. This is currently under development.

2.2.1 Explicit A-Posteriori Estimators

A way which is straightforward to get an error estimate from the identity (2.17) leads to the so-called *explicit* estimators (see e.g. [100] [2] [55] [96]). Here we see how to achieve them. The first step is to decompose eq. (2.17) into local contributions from each element: for all $v \in V$ it holds

$$\begin{aligned} B(e,v) &= L(v) - B(u_h, v) \\ &= \textstyle\sum_{K\in P}\left(\int_K fv dx + \int_{\partial K\cap\Gamma_N} gv dx - \int_K (\nabla u_h \cdot \nabla v) dx\right) \end{aligned} \tag{2.19}$$

where P is a *triangulation* [41] of Ω in finite elements K. Applying the Green's formula over each element, gives

$$B(e,v) = \sum_{K\in P}\left(\int_K rv dx + \int_{\partial K\cap\Gamma_N} Rv ds - \int_{\partial K/\ \Gamma_N} \frac{\partial u_h}{\partial n_K} v ds\right) \tag{2.20}$$

where r is the *interior residual*

$$r = f + \Delta u_h \qquad in K \tag{2.21}$$

and R is the *boundary residual*

$$R = g - \frac{\partial u_h}{\partial n_K} \qquad on \partial K \cap \Gamma_N \tag{2.22}$$

where n_K is the unit outward normal vector to ∂K. Each of these quantities is well defined thanks to the smoothness of the data and the regularity of the approximation u_h on each element. The contribution from the final term in (2.20) can be rewritten by observing that the (trace of the) function v is continuous along an edge shared by two elements, giving

$$B(e,v) = \sum_{K\in P}\left(\int_K rv dx + \int_{\partial K\cap\Gamma_N} Rv ds\right) - \sum_{\gamma\in\partial P\partial\Omega} \int_\gamma \left[\frac{\partial u_h}{\partial n}\right] v ds \tag{2.23}$$

where the final summation is over all the inter-element edges γ on the interior of the mesh. The quantity

$$\left[\frac{\partial u_h}{\partial n}\right] = n_K \cdot (\nabla u_h)_K + n_J \cdot (\nabla u_h)_J \tag{2.24}$$

defined on the edge γ separating elements K and J represents the jump discontinuity in the approximation of the flux. The identity (2.24) can be written more compactly by extending the definition of the boundary residual to incorporate the jump discontinuity in the flux. Therefore, on interior edges the definition (2.22) is augmented by

$$R = -\frac{1}{2}\left[\frac{\partial u_h}{\partial n}\right] \tag{2.25}$$

so that (2.23) then becomes

$$B(e,v) = \sum_{K\in P}\int_K rvdx + \sum_{\gamma\in\partial P}\int_\gamma Rvds \qquad \forall v \in V \tag{2.26}$$

where the final summation is over all the edges in the triangulation P.

The Galerkin orthogonality (2.18) may now be used as follows [2]. For given $v \in V$, let $\Pi_h v$ be the interpolant to v from the subspace V_h. Thanks to (1.24) and the identity (2.26) there holds

$$0 = \sum_{K\in P}\int_K r\Pi_h vdx + \sum_{\gamma\in\partial P}\int_\gamma R\Pi_h vds \tag{2.27}$$

and combining this with identity (2.26), it holds

$$B(e,v) = \sum_{K\in P}\int_K r(v-\Pi_h v)dx + \sum_{\gamma\in\partial P}\int_\gamma R(v-\Pi_h v)ds \qquad \forall v \in V \tag{2.28}$$

This identity plays an impotant role, directly or indirectly, throughout a-posteriori error analysis. It may be used to derive the a-posteriori error estimate as follows. Applying the Cauchy-Schwarz Inequality, it holds

$$B(e,v) \le \sum_{K\in P}||r||_{L_2(K)}||v-\Pi_h v||_{L_2(K)} + \sum_{\gamma\in\partial P}||R||_{L_2(\gamma)}||v-\Pi_h v||_{L_2(\gamma)} \tag{2.29}$$

Let $\tilde{K}$ denote the subdomain consisting of elements sharing a common edge with element K

$$\tilde{K} = int\left\{\cup J \in P : \bar{J}\cap\bar{K} \neq 0\right\} \tag{2.30}$$

It may then be shown [43] that there exists a constant C which is independent of v and h_K such that

$$||v-\Pi_h v||_{L_2(K)} \le Ch_K|v|_{H^1(K\ )} \tag{2.31}$$

and

$$||v - \Pi_h v||_{L_2(\partial K)} \leq C h_K^{1/2} |v|_{H^1(K\ )} \tag{2.32}$$

where h_K is the diameter of the element K. Inserting these estimates in inequality (2.29) and grouping all the contributions to form $|v|_{H^1(\Omega)}$, leads to

$$B(e,v) \leq C|v|_{H^1(\Omega)} \left\{ \sum_{K\in P} h_K^2 ||r||^2_{L_2(K)} + \sum_{\gamma\in\partial P} h_K ||R||^2_{L_2(\gamma)} \right\}^{1/2} \tag{2.33}$$

Finally, noting that $|v|_{H^1(\Omega)} \leq ||v||_E$ (this holds for more general bilinear forms $a(\cdot,\cdot)$; here it holds even with the equal sign) and substituting e in place of v, gives the a-posteriori error estimate

$$||e||_E^2 \leq C \left\{ \sum_{K\in P} h_K^2 ||r||^2_{L_2(K)} + \sum_{\gamma\in\partial P} h_K ||R||^2_{L_2(\gamma)} \right\} \tag{2.34}$$

where $||.||_E$ is the energy norm (1.26). Apart from the constant C, all of the quantities on the right-hand side can be computed explicitly from the data and the finite element approximation. Typically, the terms on the right-hand side are regrouped as follows:

$$||e||_E^2 \leq C \sum_{K\in P} \left\{ h_K^2 ||r||^2_{L_2(K)} + \frac{1}{2} h_K ||R||^2_{L_2(\partial K)} \right\} \tag{2.35}$$

The purpose in doing so is that defining the local error indicator by η_K (see the beginning of the Chapter) on element K by

$$\eta_K^2 = h_K^2 ||r||^2_{L_2(K)} + \frac{1}{2} h_K ||R||^2_{L_2(\partial K)} \tag{2.36}$$

allows one to identify contributions from each of the elements. It is then assumed that each of these quantities is a measure of the *local discretization error* over each element. In this way one can use η_K as a basis for guiding local mesh refinements.

It is broadly accepted that, given a level of required accuracy, the optimal mesh is that for which the error indicators η_K, based on the use of residuals and jumps, are equally distributed among the elements.

Besides the simplicitly in their construction, explicit error estimators suffer two major drawbacks. The first one is that they lead to local error indicators containing generic constants. These constants are, in general, unknown. In practice, one can try to find suitable bounds on the values of the constants. However, the values of constants would be indicated by the worst case scenario and therefore would usually give pessimistic estimators.

A different approach which (at least partially) overcomes this drawback is given by C.Johnson (see [55] and references therein). Here the constants (called *strong-stability factors*) are directly estimated by solving an auxiliary dual problem, instead of obtaining bounds for them.

Let us consider a general differential problem, e.g. (1.7). The method starts [55] by forming a representation of the error $u - u_h$ through the solution to a dual problem:

$$||u - u_h||^2 = (u - u_h, u - u_h) = (u - u_h, A^*\phi) = (Ae, \phi) = (f - Au_h, \phi) \tag{2.37}$$

where $f - Au_h$ is the *residual* of the strong form of the differential problem and ϕ solves the dual problem

$$A^*\phi = u - u_h \tag{2.38}$$

with A^* denoting the adjoint of A. The Galerkin orthogonality, that can be stated as

$$(Ae, v) = (f - Au_h, v) = 0, \qquad \forall v \in V_h \tag{2.39}$$

permits to transform the expression of the error as

$$||u - u_h||^2 = (f - Au_h, \phi - \pi_h\phi) \tag{2.40}$$

where $\pi_h\phi \in V_h$ is the interpolant of ϕ. Through the standard interpolation estimate

$$||h^{-\beta}(\phi - \pi_h\phi)|| \leq C_i||\nabla^\beta\phi|| \tag{2.41}$$

where C_i is an interpolation constant, h is a measure of the size of the discretization, and $\nabla^\beta\phi$ denotes derivatives of order β of the dual solution ϕ. Such estimates follow from classical interpolation theory when the solution is smooth. From a classical strong-stability estimate for the dual continuous problem

$$||\nabla^\beta\phi|| \leq S_c||u - u_h|| \tag{2.42}$$

where S_c is a strong stability factor. Applying the Schwarz Inequality we obtain

$$||u - u_h||^2 = (f - Au_h, \pi_h\phi - \phi) \leq S_cC_i||h^\beta(f - Au_h)|| \cdot ||u - u_h|| \tag{2.43}$$

which gives the following a-posteriori error estimate

$$||u - u_h|| \leq S_cC_i||h^\beta(f - Au_h)|| \tag{2.44}$$

The stability factor S_c depends in general on the particular solution being approximated, since it is defined in terms of the linearized continuous problem. In some cases, a large set of solutions have the same stability factors, e.g. for typical elliptic problems when the energy norm is used to measure the error. As previously said, analytic upper bounds on S_c often are too crude to be useful for quantitative error control. Hence, in the method of Johnson the stability factors S_c are approximated by solving the continuous dual problem numerically. The interpolation constants C_i are, as usual, determined analytically or numerically from interpolation theory, because they depend on the shape of the elements, the local order of the polynomial approximation and the choice of norms, but not on the particular solution being approximated of the mesh size.

The experiments conducted by Johnson and collegues see references in [55]) indicate that the amount of work required to compute S_c with sufficient accuracy is problem and solution dependent and depends also on the degree of reliability desired. For complex problems, complete reliability cannot be reached [81], but the degree of reliability may be increased by spending more on the computation of the S_c.

It is interesting to note that in Johnson's approach, the interplay between stability and accuracy is explicitly stated in the expression for the estimator. It indicates also that there is a trade-off between the norms used to measure stability and accuracy, i.e. using a stronger norm to measure stability allows a weaker norm to be used to measure accuracy. The goal is to balance the measurements of stability and accuracy to obtain the smallest possible bound on the error. The method of Johnson has been extende by Rannacher and collegues (see e.g. [17]): the global *stability factor* is replaced by local information from the dual solution, which is used to determine weights which are then multiplied by local residuals.

Explicit method have the advantage that the same method can be utilized to obtain a-priori and a-posteriori estimates [55] [96]. The various properties enter obviously in different forms in the a-priori and a-posteriori error estimates. The following conceptual schema clarifies in a simple way the two situations. We have, for a-priori error estimates

$$\text{small interpolation error} + \text{stability of the discrete problem} \Rightarrow \text{small error}$$

while for a-posteriori error estimates

$$\text{small residual} + \text{stability of the continuous problem} \Rightarrow \text{small error}$$

The second drawback of explicit a-posteriori error estimators is that in their expression there are two types of residual, corresponding to the interior and to the boundary. The correct relative weighting to attach to each type of residual is far from obvious. In addition, it is conceivable that there are cancellations between the two types of residual that is lost when one deals with each separately.

Implicit estimators, which are the subject of the following Section, avoid such issues by solving an auxiliary boundary value problem with e.g. the residuals as data. In this way the generic constants are avoided and the correct balance between the two types of residual is catered for by the solution process itself. As we will see, however, the drawback is that one is obliged to solve an auxiliary problem requiring an appropriate approximation scheme, and this creates its own difficulties.

2.2.2 Implicit A-Posteriori Estimators

Let us recall the error model problem (2.16). Instead of working directly on this expression to find an error bound, like was done by explicit estimators, one can consider trying to solve the residual problem characterizing the true error e:

$$\int_\Omega \nabla e \cdot \nabla v = \int_\Omega fv + \int_{\Gamma_N} gv - \int_\Omega \nabla u_h \cdot \nabla v \qquad \forall v \in V \tag{2.45}$$

By the optimality of the Galerkin method in the energy-norm (since the bilinear form which derives from the model problem here considered is symmetric) and the associated orthogonality property of the error, to obtain a non-trivial solution for this problem one must use a larger subspace than the original finite element subspace V_h. This creates the problem that the infinite dimensional space containing the error must be approximated by an appropriate finite dimensional subspace; An estimator of this kind is called *implicit* since it must be solved a refined problem to determine the error estimate. In practice, these are estimators which extrapolate approximate solutions obtained on sequences of progressively finer meshes or on sequences of meshes with shape functions of increasing order and then compare solutions to obtain an indication of the error. This definition is a little bit more general than that contained in [2].

The cost of solving the problem would be comparable with simply resolving the original problem using the finer discretization, unless special techniques are used. In the following, we see some possible ways to implement a cost-effective implicit estimator. All of them attempt to reduce the cost of solving the global problem by replacing it with a number of independent locally refined problems posed over each element in the domain. These problems could then be approximated relatively inexpensively and even solved in parallel.

2.2.2.1 Solving Local Problems

These methods [2] [100] correspond to define, for each element (or subdomain), an approximate local boundary-value problem for the error of the form

$$\int_K \nabla \phi_K \cdot \nabla v = \int_K rv dx + \int_{\partial K} v(g_K - n_K \cdot \nabla u_h) ds \qquad \forall v \in V_K \tag{2.46}$$

where r is the residual on the element interior, defined in (2.21), and g_K is an approximation of the boundary flux. The solution ϕ_K may be used as follows

$$||\phi_K||_E^2 = \int_K |\nabla \phi_K|^2 dx \tag{2.47}$$

to provide a measure of the error content in the approximation associated with element K.

This approach introduces also specific problems, namely:

- the boundary flux $n_K \cdot \nabla u$ must be approximated in some effective way

- the error residual problem may have no solution unless the condition $\int_K r dx + \int_{\partial K}(g_K - n_K \cdot \nabla u_h) ds = 0$ is satisfied

The *Subdomain Residual Method* is based on solving local problems with homogeneous essential boundary conditions on internal subdomain borders. It was formulated by Babuska and Rheinboldt [6] and it is recognized as the first a-posteriori error estimator appeared.

The *Element Residual Method* is based on solving local problems imposing Neumann boundary conditions on internal subdomain borders. The local error residual problem to be solved, becomes [2]: find $\phi_K \in V_K$ such that:

$$B_K(\phi_K, v) = F_K(v) - B_K(u_h, v) + \int_{\partial K} \left\langle \frac{\partial u_h}{\partial n_K} \right\rangle v ds \qquad \forall v \in V_K \tag{2.48}$$

where

$$\left\langle \frac{\partial u_h}{\partial n_K} \right\rangle = \begin{cases} \frac{1}{2} n_K \cdot \{(\nabla u_h)_K + (\nabla u_h)_J\}, & on \bar{K} \cap \bar{J} \\ g, & on \bar{K} \cap \Gamma_N \end{cases} \tag{2.49}$$

A major issue which arises is that this problem may have no (unique) solution. For this reason for the element residual method, particular choices of the approximating subspace V_K are made. Various spaces of bubble functions are a common choice in the literature. The basic rule for building the space V_K is to increase the order of the space used to construct the original finite-element approximation and then factor out the functions which are non-zero at the vertices of the element.

Even when there is a unique solution to the problem , this may result to be completely inconsistent with the actual error. In particular, the fact that inexact boundary data are necessarily used may give rise to spurious modes [2].

The *Equilibrated Residual Method* overcomes the problem of non-existence of the solution by choosing boundary data in such a way that the local problem is well-posed. The method is described e.g. in [1] and [2].

2.2.2.2 Updating Local Extensions

The estimator that follows this approach, which is presented in this Thesis, corresponds to define, for each element (or subdomain), an approximate global boundary-value problem for the error of the form

$$\int_\Omega \nabla \phi_K^\Omega \cdot \nabla v = \int_\Omega f v + \int_{\Gamma_N} g v - \int_\Omega \nabla u_h \cdot \nabla v \qquad \forall v \in V_K^{rloc} \tag{2.50}$$

where $V_K^{rloc} \supset V_h$ is the approximation space V_h used to get the solution u_h which has been refined locally, i.e. within element K. This creates a local *extension* (cfr. Section 2.1.1); here we will consider only h-extensions. The solution ϕ_Ω^K, which is now defined on all Ω, may be used now as follows:

$$||\phi_\Omega^K||_E^2 = \int_K |\nabla \phi_\Omega^K|^2 dx \tag{2.51}$$

to provide a measure of the error content in the approximation associated with element K, and

$$||\phi_\Omega^K||_E^2 = \int_{\Omega - K} |\nabla \phi_\Omega^K|^2 dx \tag{2.52}$$

to provide a measure of the *pollution error* (cfr. Section 2.2.5) created by the improper discretization in the region K and, more precisely,

$$||\phi_\Omega^K||_E^2 = \int_{K'} |\nabla \phi_\Omega^K|^2 dx \tag{2.53}$$

to provide a measure of the *pollution error* added to element K' by the improper discretization in the region K.

This new estimator (which is further described on Section 2.3) belongs to the class of implicit estimators since it finds an approximation for the error residual problem, but it is conceptually different from the existing implicit estimators since it solves global locally-refined problems instead of local problems. It is more adherent to the paradigm of the *multi-scale variational analysis* [78] (i.e. the approximation error is governed by Green's functions driven by the residuals) than to the usual paradigms of a-posteriori error estimation (i.e. the error is bounded by the norm of the residuals).

With this estimator are avoided the previously mentioned drawbacks of explicit and implicit estimators.

2.2.3 The Zienkiewicz-Zhu Method

This conceptually simple error estimator does not make use of the informations of the underlying differential problem. Its simplicity, but also its reliability not much different from that owned by the other estimators, has given wide popularity to this estimator, specially among the engineering community. Behind its success in practical use, the mathematicians have demonstrated that this estimator retains some fundamental properties (see e.g. [91]) and numerical comparisons have also assessed its robustness (see e.g. [8]).

With this method, a-posteriori error estimates are obtained comparing two approximations for the gradient of the solution having different approximation orders. More precisely, if Gu_h is a better approximation of ∇u than that obtainable from u_h directly, i.e. ∇u_h:

$$||\nabla u - Gu_h||_{L^2} \leq \beta ||\nabla u - \nabla u_h||_{L^2}$$

then, if Gu_h is also easy to compute, we can use $||Gu_h - \nabla u_h||_{L^2}$ as an error estimator of the gradient of the solution. It turns out in practice that the quality of the *recovered derivative* Gu_h, as one can imagine, makes the difference in the quality of the error estimate. For this reason, the first formulation of the *ZZ method* [106] has been replaced by its "superconvergent" version. The superconvergence of the derivatives of a finite element solution and in particular the robustness of this type of estimator have been considerably discussed in the literature (see e.g. [12] [104]).

Here we present the '87 method, which was presented for linear elasticity problems of structural mechanics. Given

$$N = [N_1, \dots, N_n] \qquad \text{nodal shape functions}$$

$$U = \begin{bmatrix} U_1 \\ \vdots \\ U_n \end{bmatrix} \qquad \text{nodal computed solution values}$$

$$u_h = NU \qquad \text{computed solution function}$$

$$\hat{\sigma}_i = \tfrac{\partial N}{\partial x_i}\, U \quad (i = 1, 2, 3) \qquad \text{computed solution derivatives}$$

A better approximation for the gradient of the computed solution can be obtained using the same nodal functions used for the computation of the solution and imposing the Galerkin orthogonality condition for finding a continuous piecewise linear approximation σ^* of the solution gradient:

$$\int_\Omega N^T \left(\bar{\sigma}_i^* - \hat{\sigma}_i\right) \mathrm{d}\Omega = 0 \qquad (i = 1, 2)$$

that is

$$\begin{aligned} \int_\Omega \left[N^T N \sigma_i^* - N^T \frac{\partial N}{\partial x_i}\, U\right] \mathrm{d}\Omega = 0 \\ \left[\int_\Omega N^T N \mathrm{d}\Omega\right] \sigma_i^* = \left[\int_\Omega N^T \frac{\partial N}{\partial x_i} \mathrm{d}\Omega\right] U \\ M\sigma_i^* = Q_i U \end{aligned} \tag{2.54}$$

Usually the matrices M and Q have been previously calculated for computing the solution, but it still remains a linear system comparable to that which gave the solution values U_i, to solve. For this reason, in the practical application of the method, the author suggest a "lumping" procedure for the matrix M and the system becomes diagonal. Note that this lumping procedure corresponds to go from the 'L_2-projection to the local-averaging of the gradient [91].

Define

$$\epsilon_i = \sigma_i^* - \hat{\sigma}_i = N\sigma_i^* - \frac{\partial N}{\partial x_i}\, U \qquad (i = 1, 2).$$

and the a-posteriori error estimates can be expressed as the sum of each element contribution

$$\|\epsilon\|_\Omega^2 = \int_\Omega \epsilon^T \epsilon \, \mathrm{d}\Omega = \sum_{k=1}^m \left[\int_{e_k} \epsilon^T \epsilon \, \mathrm{d}e_k\right] = \sum_{k=1}^m \|\epsilon\|_{e_k}^2$$

where e_k is the k-th element of the mesh.

In the '92 method, called the *Superconvergent Patch Recovery (SPR)* technique, has been changed the gradient recovery mechanism: instead of the L_2-projection (which in practice was implemented as a simple local averaging of the gradient), it has been introduced a local least-squares approximation, taken from few discrete points in the patch. The optimality arises if these are points of superconvergence.

2.2.4 Equivalence to the Error

A major property demanded to all successful error estimators $\eta = \left\{\sum_{K\in P} {\eta_K}^2\right\}^{1/2}$ is that positive constants C_1 and C_2 exist such that

$$C_1||e|| \leq \eta \leq C_2||e|| \tag{2.55}$$

where $||e||$ is the global error in the norm chosen. in this way η tends to zero at the same rate as the true error. This property is called *equivalence* of the estimator, i.e. the estimator is an equivalent measure of the error.

Recalling the definition of the *effectivity index* (2.4), we can say also that an estimator is *equivalent* to the error if its effectivity index is bounded above and below by two strictly positive constants depending on the regularity of the mesh but not on the mesh size.

Explicit local error indicators (2.36) and the corresponding global error estimators can be demonstrated to be equivalent to the error, i.e. [2] [99] [100]:

THEOREM: *"Let η_K denote the local error indicator*

$$\eta_K^2 = h_K^2||r||^2_{L_2(K)} + \frac{1}{2}h_K||R||^2_{L_2(\partial K)}$$

where r and R are the interiori and boundary residuals. Then there exists a constant C depending only on the shape regularity of the elements such that

$$\frac{1}{C}||e||_E^2 \leq \sum_{K\in P} \eta_K^2 \leq C\left\{||e||_E^2 + \sum_{K\in P} h_K^2||f-\Pi_p f||^2_{L_2(K)} + \sum_{\gamma\subset\Gamma_N} h_K||g-\Pi_p g||^2_{L_2(\gamma)}\right\} \quad .$$

Moreover, the local bound

$$\frac{1}{C}||e||_E^2 \leq \sum_{K\in P} \eta_K^2 \leq C\left\{||e||_E^2 + \sum_{K\in P} h_K^2||f-\Pi_p f||^2_{L_2(K)} + \sum_{\gamma\subset\Gamma_N} h_K||g-\Pi_p g||^2_{L_2(\gamma)}\right\}$$

is valid.

Note that, in case of triangular elements, the requirement of shape-regularity of the elements corresponds to the well-known *minimum angle condition.*

For *implicit* estimators, the proof of the equivalence can be made by relating them to the explicit ones [2]. By applying the Green's formula to the right-hand-side of the error residual equation (2.46) one obtains:

$$B_K(\phi_K, v) = \int_K rvdx + \int_{\partial K} Rvds \qquad \forall v \in V_K \tag{2.56}$$

where r and R are the residuals (2.21) (2.22) appearing in the explicit error estimators. Choosing the function v to be the estimated error function ϕ_K leads to

$$\eta_K^2 = \int_K r\phi_K dx + \int_{\partial K} R\phi_K ds \qquad . \tag{2.57}$$

Since the functions in V_K vanish at the vertices of K, then standard scaling arguments [21] imply that there is a constant C which depends on the shape-regularity of the element K but is independent from its size, such that

$$||\phi_K||_{L_2(K)} \leq Ch_K|\phi_K|_{H^1(K)} \tag{2.58}$$

and for any edge $\gamma \subset \partial K$

$$||\phi_K||_{L_2(\gamma)} \leq Ch_K^{1/2}|\phi_K|_{H^1(K)} \tag{2.59}$$

With these results and the Cauchy-Schwarz inequality, it holds

$$\eta_K^2 \leq C\left\{h_K||r||_{L_2(K)} + \sum_{\gamma\subset\partial K} h_K^{1/2}||R||_{L_2(\gamma)}\right\}|\phi_K|_{H^1(K)} \tag{2.60}$$

from which one can deduce that

$$\eta_K \leq C\left\{h_K^2||r||^2_{L_2(K)} + \frac{1}{2}\sum_{\gamma\subset\partial K} h_K||R||^2_{L_2(\gamma)}\right\}^{1/2} \tag{2.61}$$

where the term on the right-hand-side is the *explicit* error estimator (2.36).

The equivalence of the method presented in this Thesis, based on updating local *extensions*, derives directly from the assumed convergence of the finite element solution. In fact, assuming that a series of increasing *extensions* converges monotonically in norm to the true solution with a rate-of-convergence p, it derives that the difference between two terms of the series (which is taken as the error estimate) is bounded above by the true error and it is bounded below by a fraction of the true error which depends on p.

The equivalence of the *Zienkiewicz-Zhu method* has been proved in [91] by demonstrating the equivalence with some explicit estimators, whose equivalence to the error was previously demonsatrted.

2.2.5 Pollution Error Estimation

In the analysis of local error estimation in finite element methods made by Wahlbin [101], the approximation error is formed by the sum of two components, one measuring local approximability and the second measuring global influences. When global influences dominate over the first component it is said that there is *pollution* effects.

In a more practical definition due to Babuska [11], given a subdomain of interest ω_h, the *local-error* is the error in the finite element solution of the original problem in ω_h with Dirichlet boundary conditions equal to u on the boundary of ω_h. The *pollution-error* is the exact solution of the original problem with Dirichlet boundary conditions equal to $(u - u_h)$ on the boundary of ω_h.

The presence of the pollution error then means that the accuracy of the approximate solution in a subdomain of interest depends also on the accuracy in the remaining part of the domain. When the mesh is refined in a subdomain of interest, the error of the local best approximation is reduced but the pollution-error is not affected. The pollution error is an error that have a precise local origin and spreads out through the rest of the domain. Two things then characterize the pollution error: sources and distribution mechanisms.

The recognized causes of the pollution-error creation are many. Singularities introduced by rough domains (usually domains with corner points) and abrupt changes in the boundary conditions are the most studied up-to-date [101] [9] [97], but also other situations, like problems governed by the Helmholtz equation with large wavenumber [13] [14].

About the error-distribution mechanisms, there are only few contributions. For a practical computation of the pollution error see e.g. [11] and for a theoretical framework which describes the error distribution see e.g. [76] [77] [78].

Pollution error estimation is a major issue in adaptivity, since it turns out [9] that if the pollution error in a subdomain is large, nothing can be said about the reliability of any error estimator which is based on local computations in the subdomain. Since all existing error estimators are based on local computations, they cannot detect the pollution-error.

An important fact about pollution-error is that [9] *it is possible to control the pollution-error in any interior mesh-cell by proper design of the mesh.* This means that in principle with mesh adaptation it is possible to control also the pollution error. If this is also done in practice, then it is possible to ensure the reliability of the error estimators in any region of interest.

At the present state of the knowledge, two are the possibilities of controlling the pollution error. The first is to equilibrate the mesh in the norm of interest and the second is to estimate the pollution error.

In [9] it is shown e.g. that when the mesh is nearly-equilibrated in the energy norm over the entire domain, the pollution-error in the derivatives is negligible. For some applications (e.g. large scale) this could be too much expensive and not efficient, if the region of interest to the adaptive analysis is small compared to the entire domain.

The estimation of the pollution error requires additional mechanisms to the existing a-posteriori error estimators. We see an example [11], who developes a pollution error estimator for the *element residual method*.

2.3 The Local-Extensions-Update Method

This *implicit* estimator, which has been presented in Section 2.2.2.2, avoids the drawbacks of the existing implicit estimators by making the assumption that the computational cost required to update the solution after a local mesh refinement is much less than the cost which have been spent to obtain the solution itself. In Section 2.3.2 this assumption is tested experimentally on some 2D sample problems. The computational costs which are shown, refer to the case of updating only the variables whose update is non-trivial.

In the third Part of this Thesis it is presented a linear systems solver which is designed to update efficiently a finite element solution after a local mesh refinement. The target of this solver is to update only the variables which change significantly (i.e. to satisfy the assumption just made).

In this way, the *local extensions update* method obtains an error estimate by comparing two solutions obtained using in a small subdomain two different discretization steps. This is analogous to what is commonly done for ODE; this approach would be extremely expensive for PDE without an appropriate solver.

The advantages of this method are that spurious modes and difficulties on imposing the local boundary conditions are simply avoided. Moreover, the *pollution error* can be directly estimated, while for existing methods, which are based on local computations, additional algorithms are required (see e.g. [11]).

Finally, the accuracy of the estimates can be improved quite easily by the adoption of extrapolation techniques.

This method can be used with h-, p- or *hp-extensions*.

2.3.1 Error Estimation Procedure

The estimation is made in two steps. The first step is to control the *pollution error* to an appropriate level, in the region where local error estimates are desired; the aplitude of this region varies considerably from application to application (see e.g. [9]). The second step is to make the local estimate of the approximation error.

The first step is the most computationally intensive, since the corresponding update is greatly non-local. For this problem, a careful implementation of the solver (see Part III) must be adopted. The local error estimate, instead, corresponds also to a local computation.

As already mentioned, the local error estimate can be improved by extrapolation techniques, see e.g. [27] for an example of Richardson extrapolation on two grids. Extrapolation

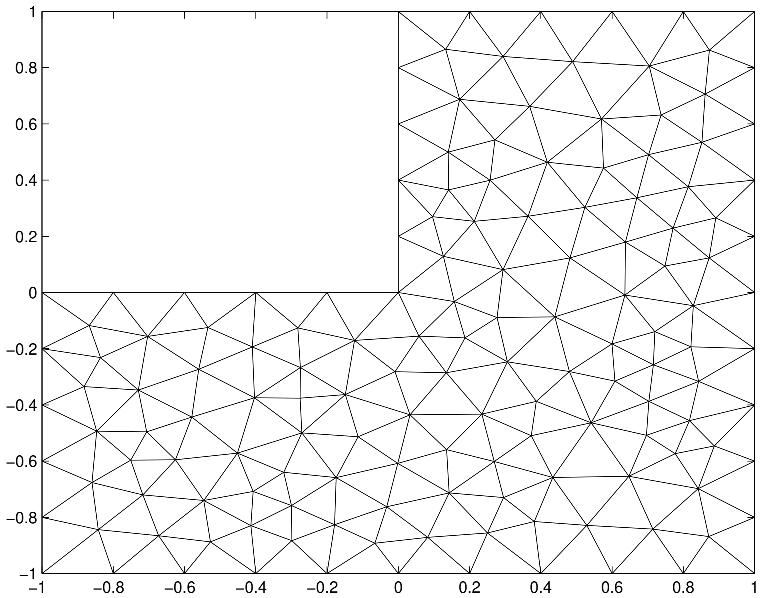

Figure 5: L-shaped domain discretization using linear triangular finite-elements

methods succeed if we are in the asymptotic range; in this case they allow to better estimate the true error without refinining the mesh too much, which is numerically dangerous other than expensive.

2.3.2 Experimental Results

The numerical experiments necessary to assess the convenience of using the *local extensions update* method must measure the minimum cost required by algebraic computations to make the update. Then, in the third Part of this Thesis, it will be shown how much close to this minimum the solver proposed is.

Let us consider the following problem:

$$-\Delta u = 10 \quad in \quad \Omega$$

where $\Omega \in R^2$ is an L-shaped domain which has been discretized using linear triangular elements (see Figure 5) and the following boundary conditions are imposed: homogenoeus Dirichlet boundary conditions on all the edges, except the two edges which form the reentrant corner. On this two edges, various types of discontinuous boundary conditions are imposed:

A) homogeneous Dirichlet on one edge and non-homogenous Dirichlet (with values from 0 to 10) on the other.

B) homogeneous Neumann on one edge and non-homogenous Neumann (with values from 0 to 10) on the other.

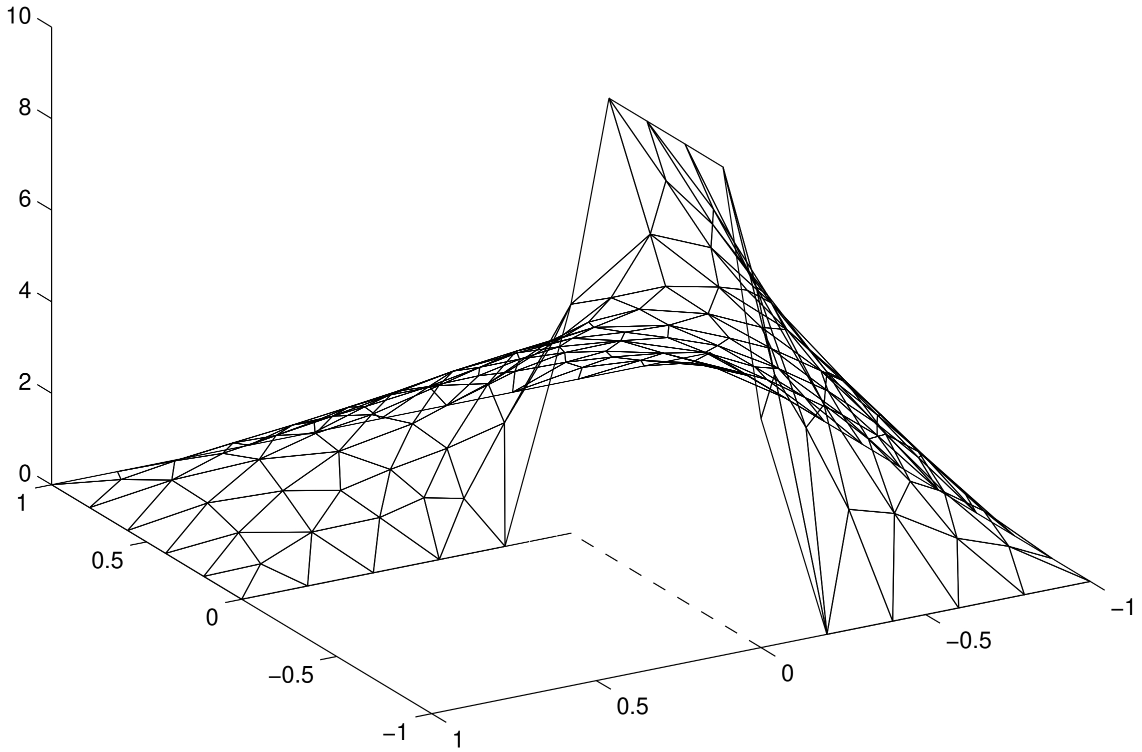

Figure 6: solution with boundary conditions of type “A”

C) homogeneous Dirichlet on one edge and non-homogenous Neumann (with values from 0 to 10) on the other.

D) homogeneous Neumann on one edge and non-homogenous Dirichlet (with values from 0 to 10) on the other.

Note that this type of domain and boundary conditions are sources of *pollution error* [101]. For a characterization of the singularities produced, see [7] [9].

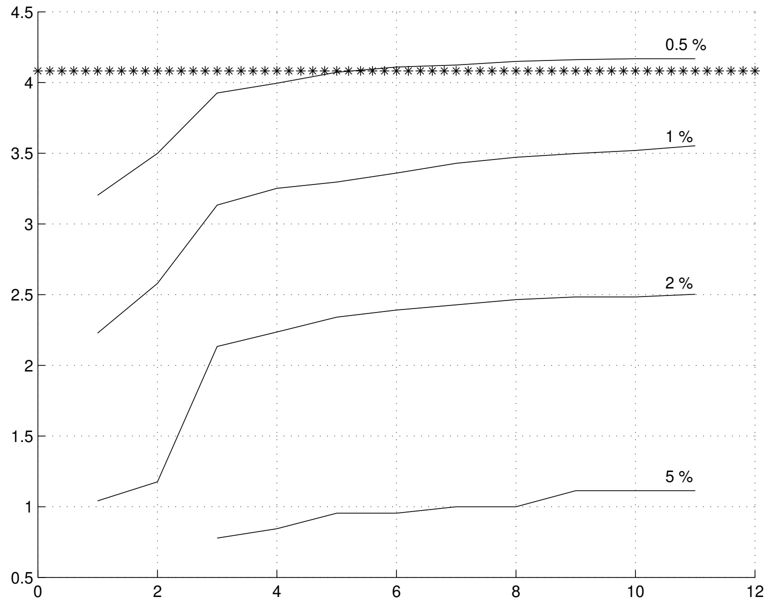

Figure 7: computational cost (in base-10 logarithmic scale) of the pollution error elimination with an $O(N^2)$ solver, versus the amplitude of the non-homogeneous boundary condition for case “A”

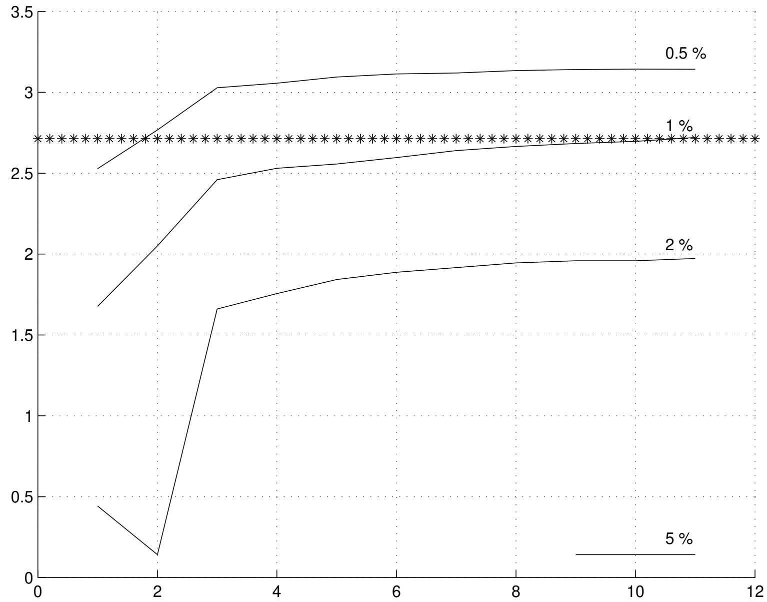

Figure 8: computational cost (in base-10 logarithmic scale) of the pollution error elimination with an $O(N \cdot ln(N))$ solver, versus the amplitude of the non-homogeneous boundary condition for case “A”

In the following graphics, the computational cost (in base-10 logarithmic scale) of the

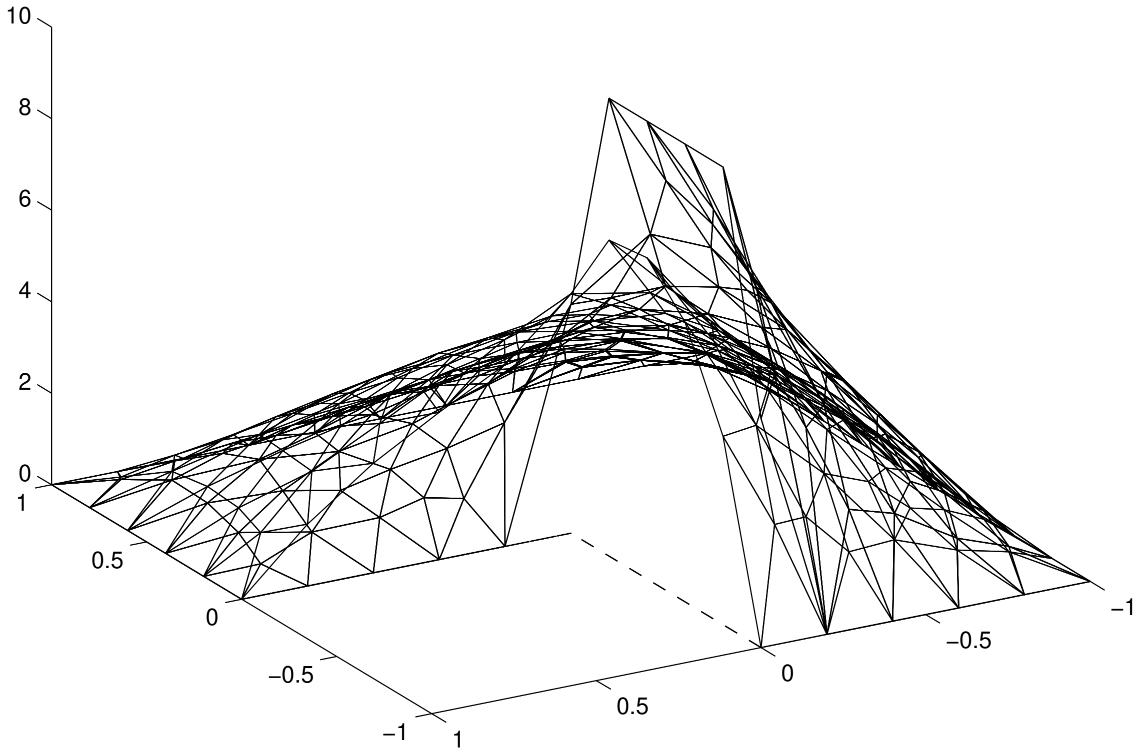

Figure 9: solution with boundary conditions of type “B”

pollution error elimination, versus the amplitude of the non-homogeneous boundary condition, is shown.

Two different computational complexities for the solver are assumed: N^2 and $NlogN$, which are representative of the most part of existing sparse systems solvers [4].

To eliminate the pollution, each triangle is properly refined and the solution updated. For the update problem, only the variables whose update is significant above a certain percentage, are considered. In the graphics are presented the costs corresponding to $5, 2, 1, 0.5\%$ tolerances, which are usually the most restrictive used by the finite-elements community in the applications. The missing curves are identically null curves.

With an horizontal line of asterisks is indicated the corresponding computational cost necessary to obtain the finite element solution on the mesh of figure 5.

.

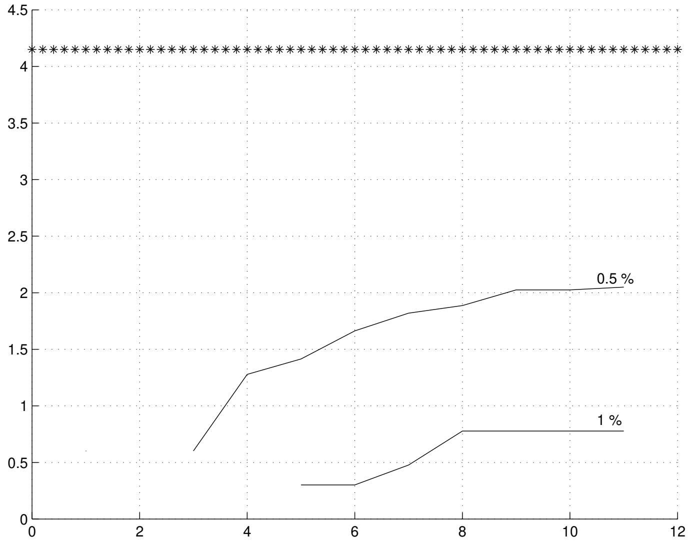

Figure 10: computational cost (in base-10 logarithmic scale) of the pollution error elimination with an $O(N^2)$ solver, versus the amplitude of the non-homogeneous boundary condition for case "B"

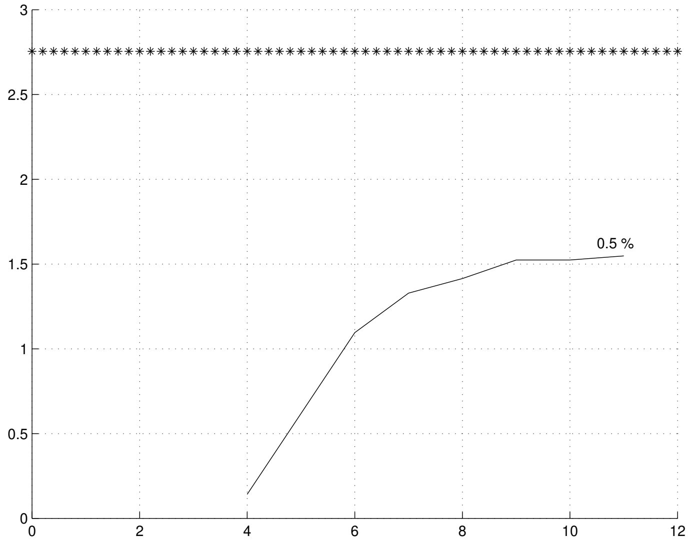

Figure 11: computational cost (in base-10 logarithmic scale) of the pollution error elimination with an $O(N \cdot ln(N))$ solver, versus the amplitude of the non-homogeneous boundary condition for case "B"

.

.

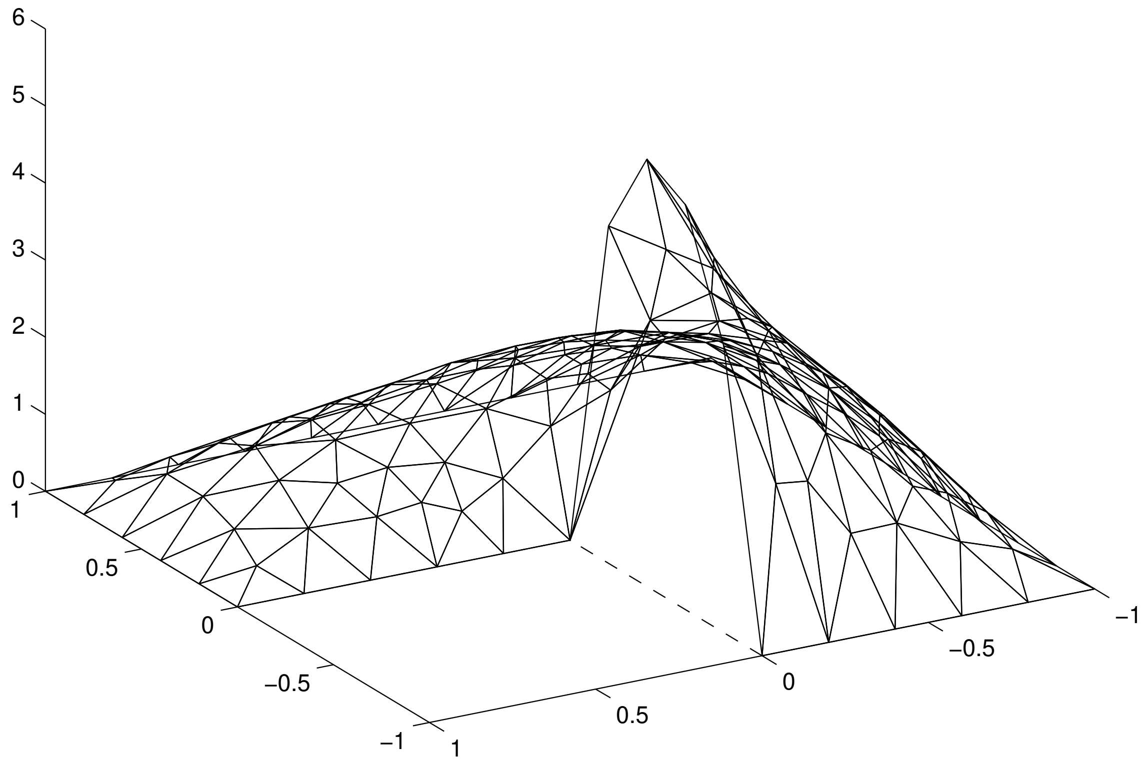

Figure 12: solution with boundary conditions of type "C"

.

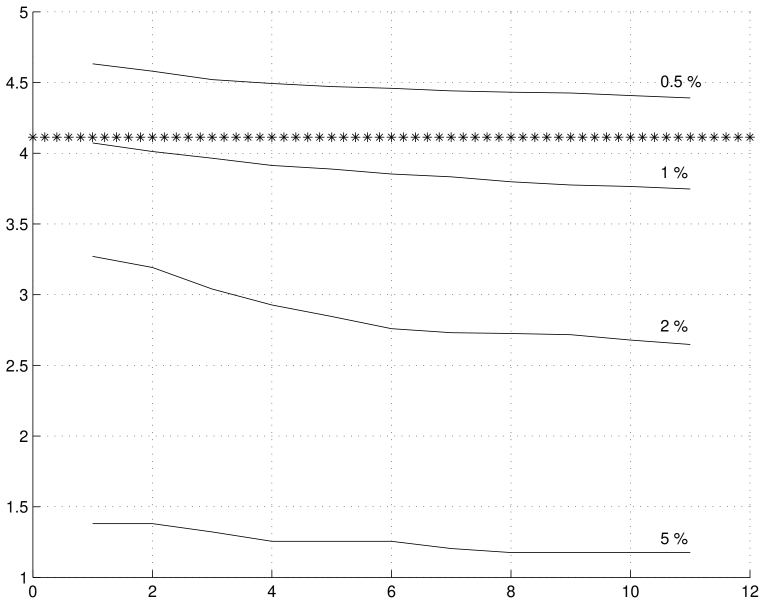

Figure 13: computational cost (in base-10 logarithmic scale) of the pollution error elimination with an $O(N^2)$ solver, versus the amplitude of the non-homogeneous boundary condition for case "C"

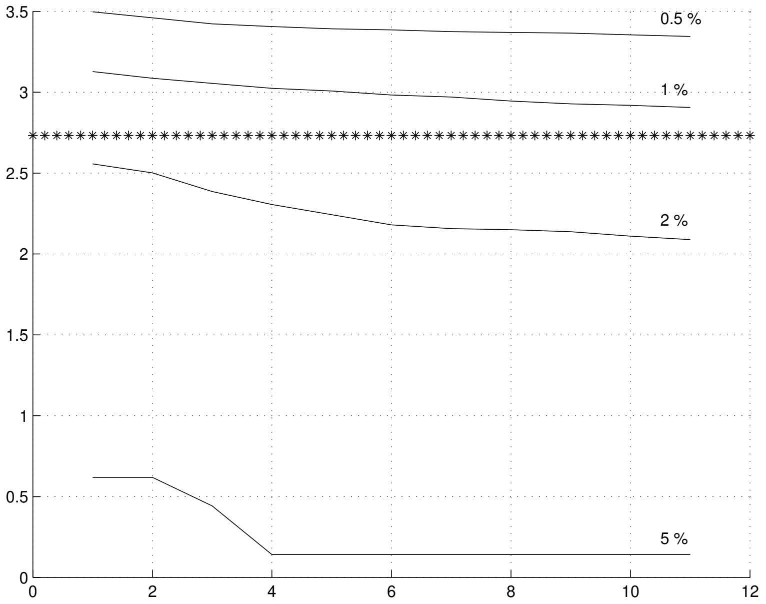

Figure 14: computational cost (in base-10 logarithmic scale) of the pollution error elimination with an $O(N \cdot ln(N))$ solver, versus the amplitude of the non-homogeneous boundary condition for case "C"

.

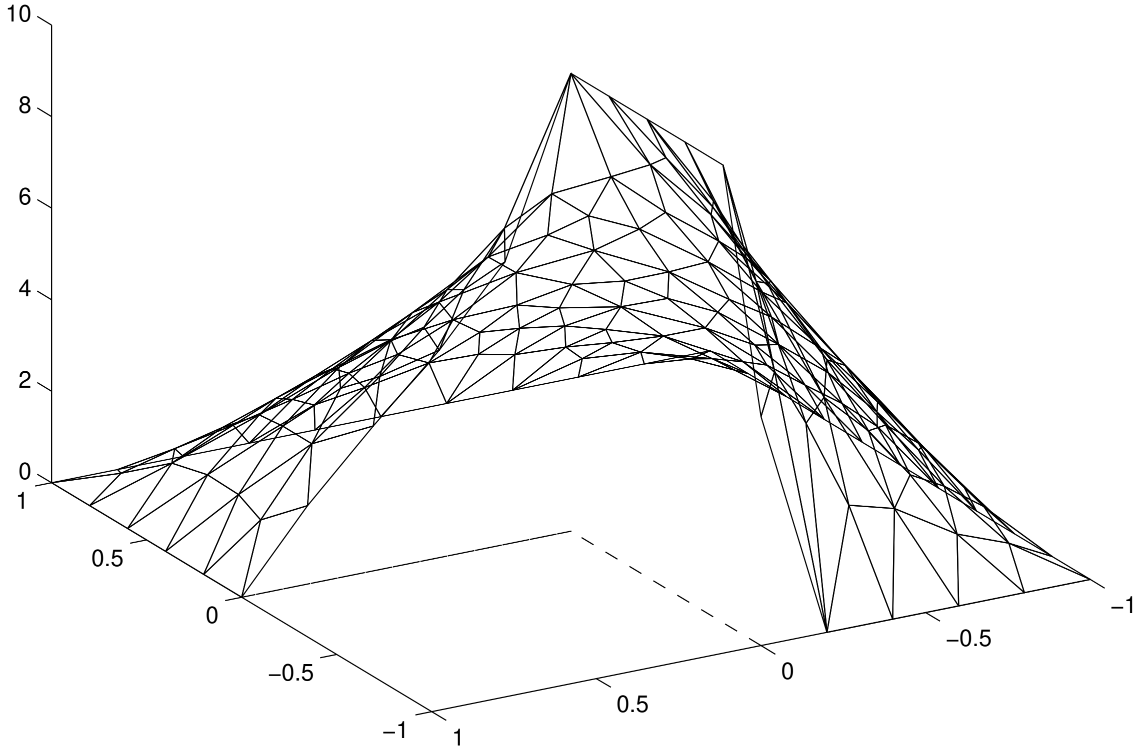

Figure 15: solution with boundary conditions of type “D”

.

.

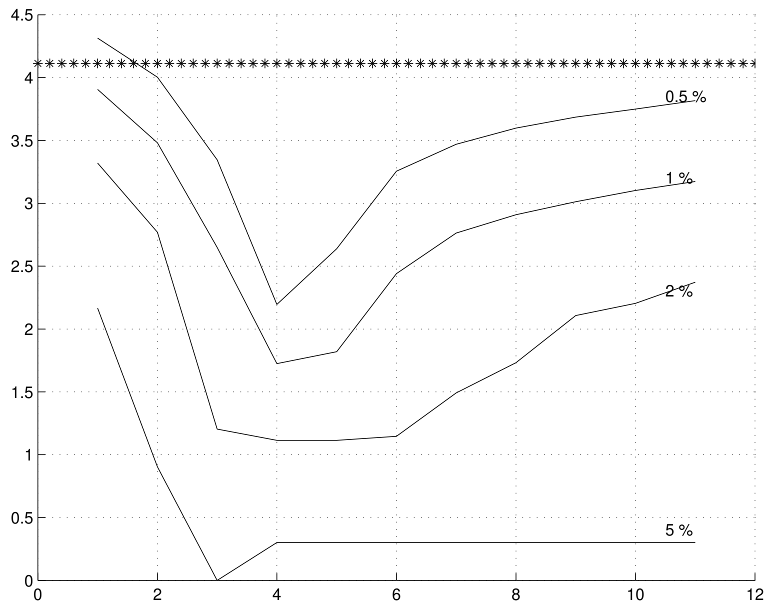

Figure 16: computational cost (in base-10 logarithmic scale) of the pollution error elimination with an $O(N^2)$ solver, versus the amplitude of the non-homogeneous boundary condition for case "D"

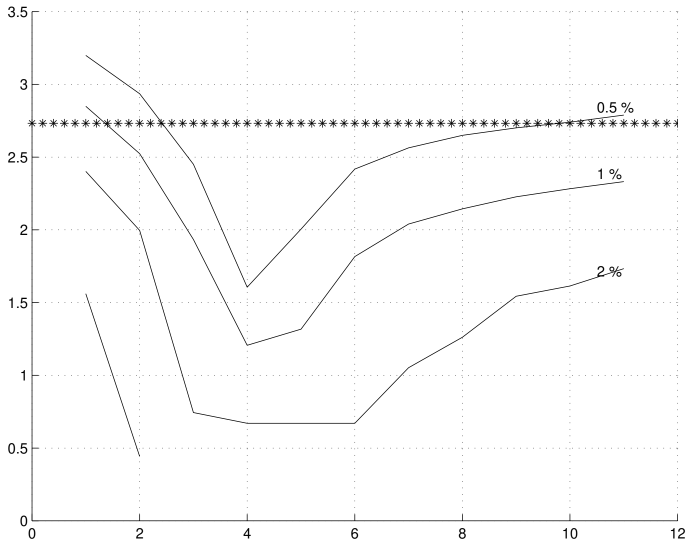

Figure 17: computational cost (in base-10 logarithmic scale) of the pollution error elimination with an $O(N \cdot ln(N))$ solver, versus the amplitude of the non-homogeneous boundary condition for case "D"

Chapter 3

3
Error Estimates for Time-Dependent Problems

Traditionally, the most commonly used approach to solve time/dependent problems with the Finite Element Method is the so called *semi-discretization* method. It consists of a prior finite element discretization in space that produces a system of ordinary differential equations in time, and a solver for this system of ODEs. For time dependent problems, some discretization schemes in time have been developed ad hoc for finite elements: the Newmark algorithm, the predictor/multi-corrector algorithm, ... (see e.g. [73] for a description of these methods).

A successful alternative to semi-dicretization methods consists in performing a space-time finite element discretization (i.e. each element is defined on a portion of the space-time domain), in which the space-time domain is sectioned in *slabs* each of which consists of the entire space-domain and a time-interval. The continuity between space-time *slabs* can be imposed strongly (leading to the *Continuous Galerkin* method in time) or weakly (leading to the *Discontinuous Galerkin* method in time). The Continuous Galerkin method brings to an explicit solver and can suffer of conditional stability, while the Discontinuous Galerkin brings to an implicit solver and is therefore unconditionally stable, more expensive (the unknowns at least double) and higher-order accurate (see the literature).

The Continuous Galerkin has been applied to elastodynamics, heat conduction and advective-diffusive systems, see e.g. [28] and [30] for linear parabolic problems, [33] for quasi-linear parabolic problems and [32] for non-linear parabolic problems.

The Discontinuous Galerkin (for an introduction see e.g. [80]) has been originally developed for first-order hyperbolic equations, successively applied to problems in incompressible and compressible fluid dynamics and heat conduction, and also to second-order hyperbolic equations [75] [74]. Further references on both methods can be found in [75].

3.1 Adaptive Finite-Elements formulations for Time-Dependent Problems

For all of the methods previously mentioned, the adaptivity consists in modifying the discretization in space and choosing the width of the next time-step (respectively, space-time slab). In space, mesh refinement/un-refinement is made locally (as happens for stationary problems) while in time the selection of the depth of the slab is usually made globally with respect to space.

3.1.1 Semi-Discretization Examples

The adaptivity for semi-discretization methods is usually a straightforward implementation of adaptive algorithm for stationary finite element discretizations in space, and adaptive algorithms for ODEs in time. For example, in [103] are presented adaptive semi-discretization algorithm for elasto-dynamics and in particular an a-posteriori error estimate for the Newmark discretization scheme.

In the following we describe another adaptive semi-discretization algorithm, which is suited for Taylor-Galerkin discretizations [51] and has been applied to a shallow-waters problem [36]. The numerical study of the shallow waters equations had been previously done in [34] [29] [35]. Here we will sketch it briefly. The solution of the shallow water problem led to the following semi-linear system:

$$
\begin{aligned}
\frac{\partial \eta}{\partial t} + \frac{\partial H u_1}{\partial x_1} + \frac{\partial H u_2}{\partial x_2} &= 0 \\
\frac{\partial u_1}{\partial t} + g\frac{\partial \eta}{\partial x_1} &= f u_2 - \frac{g\sqrt{{u_1}^2 + {u_2}^2}}{C^2 H} u_1 \\
\frac{\partial u_2}{\partial t} + g\frac{\partial \eta}{\partial x_2} &= -f u_1 - \frac{g\sqrt{{u_1}^2 + {u_2}^2}}{C^2 H} u_2
\end{aligned}
\tag{3.1}
$$

where, having performed the integration of the variables along the vertical dimension,

$$\eta = \eta(x_1, x_2) = \text{ is the height of the free surface}$$

$$u_1 = u_1(x_1, x_2) = \text{ is the velocity component along } x_1 \text{ direction}$$

$$u_2 = u_2(x_1, x_2) = \text{ is the velocity component along } x_2 \text{ direction}$$

Equation set (3.1) can be written, in vector form, as

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_1}{\partial x_1} + \frac{\partial \mathbf{F}_2}{\partial x_2} = \mathbf{R}_S \tag{3.2}$$

where $\quad \mathbf{U} = [\eta, u_1, u_2]^T \quad \mathbf{F}_1 = [H u_1, g\eta, 0]^T \quad \mathbf{F}_2 = [H u_2, 0, g\eta]^T$

and $\quad \mathbf{R}_s = \left[0 \quad , \quad f u_2 - \frac{g|u|u_1}{C^2 H} \quad , \quad -f u_1 - \frac{g|u|u_2}{C^2 H}\right]^T$

By taking $\mathbf{F}_i = \mathbf{F}_i^* + \mathbf{F}_i^{**} \quad (i = 1, 2) \qquad \mathbf{U} = \mathbf{U}^* + \mathbf{U}^{**}$ with $\mathbf{F}_i^* = 0$, $\mathbf{F}_i^{**} = \mathbf{F}_i$, system (3.2) can be splitted into

$$\frac{\partial \mathbf{U}^*}{\partial t} = \mathbf{R}_s \tag{3.3}$$

$$\frac{\partial \mathbf{U}^{**}}{\partial t} + \frac{\partial \mathbf{F}_1^{**}}{\partial x_1} + \frac{\partial \mathbf{F}_2^{**}}{\partial x_2} = \mathbf{0} \tag{3.4}$$

3.1.1.1 Discretization of equation (3.3)

Now, thanks to the particular operator-splitting adopted, equation (3.3) does not contain derivatives in space and so must be fulfilled at each point of the domain independently; in particular we will evaluate it at nodal points of the mesh used for the solution of (3.4). This corresponds to say that discretizing (3.3) in space creates a diagonal system of equations. After performing the discretization in time, the independence of the equations also simplifies considerably the stability analysis because this can be performed for each equation separately; for this reason an explicit formula for the eigenvalues ρ can be obtained and the condition $|\rho| < 1$ easily checked. Moreover, the discretization in time can be easily evaluated adopting a second order Taylor series expansion [51] as follows:

$$\mathbf{U}^{*(n+1)} = \mathbf{U}^{*(n)} + \Delta t \left(\frac{\partial \mathbf{U}^*}{\partial t}\right)^{(n)} + \frac{\Delta t^2}{2}\left(\frac{\partial^2 \mathbf{U}^*}{\partial t^2}\right)^{(n)} \tag{3.5}$$

Since $\dfrac{\partial \mathbf{U}^*}{\partial t} = \mathbf{R}_s$ and $\dfrac{\partial^2 \mathbf{U}^*}{\partial t^2} = \dfrac{\partial \mathbf{R}_s}{\partial t} = \dfrac{\partial \mathbf{R}_s}{\partial \mathbf{U}^*}\dfrac{\partial \mathbf{U}^*}{\partial t}$,
the system (3.5) can be rewritten as

$$\mathbf{U}^{*(n+1)} = \mathbf{U}^{*(n)} + \Delta t(\mathbf{R}_s)^{(n)} + \frac{\Delta t^2}{2}(\mathbf{G}\mathbf{R}_s)^{(n)} \qquad \text{where} \qquad \mathbf{G} = \frac{\partial \mathbf{R}_s}{\partial \mathbf{U}^*}. \tag{3.6}$$

and solved doing straightforward calculations for evaluating $(\mathbf{R}_s)^{(n)}$ and $(\mathbf{G}\mathbf{R}_s)^{(n)}$. This discretization will be advantageous for obtaining a-posteriori error estimates for the time-discretization error.

3.1.1.2 Discretization of equation (3.4)

For the discretization of equation (3.4) it is convenient to write it in the extended form

$$\begin{cases} \dfrac{\partial}{\partial t}\left(\eta^{**}\right) + \dfrac{\partial}{\partial x_1}\left(H u_1\right) + \dfrac{\partial}{\partial x_2}\left(H u_2\right) = 0 \\ \dfrac{\partial}{\partial t}\left(u_1^{**}\right) + \dfrac{\partial}{\partial x_1}\left(g\eta\right) = 0 \\ \dfrac{\partial}{\partial t}\left(u_2^{**}\right) + \dfrac{\partial}{\partial x_2}\left(g\eta\right) = 0 \end{cases} \tag{3.7}$$

This is discretized in time according to the θ-method proposed in [98], giving

$$\begin{cases} (\Delta\eta^{**})^{(n+1)} + \Delta t \left[\dfrac{\partial}{\partial x_1}\left(H u_1^{(n+\theta_1)}\right) + \dfrac{\partial}{\partial x_2}\left(H u_2^{(n+\theta_1)}\right)\right] = 0 \\ (\Delta u_1^{**})^{(n+1)} + \Delta t \dfrac{\partial}{\partial x_1} p^{(n+\theta_2)} = 0 \\ (\Delta u_2^{**})^{(n+1)} + \Delta t \dfrac{\partial}{\partial x_2} p^{(n+\theta_2)} = 0 \end{cases} \tag{3.8}$$

where $p = g\eta$, θ_1 and θ_2 are real parameters in $[0,1]$, $(\Delta\eta^{**})^{(n+1)} = \eta^{**(n+1)} - \eta^{**(n)}$, $(\Delta u_i{}^{**})^{(n+1)} = u_i{}^{**(n+1)} - u_i{}^{**(n)}$ and

$$\begin{cases} u_i^{(n+\theta_1)} = u_i^{(n)} + \theta_1 \left[(\Delta u_i^*)^{(n+1)} + (\Delta u_i^{**})^{(n+1)} \right] & i = 1, 2 \\ p^{(n+\theta_2)} = p^{(n)} + \theta_2 (\Delta p^{**})^{(n+1)} \end{cases} \tag{3.9}$$

After some manipulation and discretizing in space using linear triangular finite elements, we have:

$$\begin{aligned} &\left(\tfrac{1}{g}\mathbf{M} + \Delta t^2 \theta_1 \theta_2 \mathbf{S}\right) (\Delta p^{**})^{(n+1)} = \\ &\qquad - \Delta t \left\{ \textstyle\sum_{i=1}^{2} \mathbf{Q}_i \left[H \left(u_1^{(n)} + \theta_1 \left(\Delta u_i^* \right)^{(n+1)} \right) \right] + \Delta t \theta_1 \mathbf{S} p^{(n)} \right\} \end{aligned} \tag{3.10}$$

$$\mathbf{S} = \sum_{i=1}^{2} \left(\int_\Omega \frac{\partial[\phi]}{\partial x_i} H \frac{\partial[\phi]^T}{\partial x_i} d\Omega \right) \qquad \text{and} \qquad \mathbf{Q}_i = \int_\Omega [\phi] \frac{\partial[\phi]^T}{\partial x_i} d\Omega. \tag{3.11}$$

where ϕ is the vector of shape functions at all nodes in the mesh. Once the increment in pressure has been evaluated from (3.10), it can be used for the calculation of $(\Delta u_i^{**})^{(n+1)}$:

$$\mathbf{M}(\Delta u_i^{**})^{(n+1)} = -\Delta t \mathbf{Q}_i \left[p^{(n)} + \theta_2 (\Delta p^{**})^{(n+1)} \right] \qquad i = 1, 2. \tag{3.12}$$

where $\mathbf{M}_i = \int_\Omega [\phi][\phi]^T d\Omega$, in order to obtain the velocity sub-increments $(\Delta u_1^{**})^{(n+1)}$ and $(\Delta u_2^{**})^{(n+1)}$.

The algorithm for evaluating $(\Delta U^{**})^{(n+1)}$ is unconditionally stable for suitable choices of θ_1 and θ_2 [98] and can run at higher time-steps than the algorithm that computes $(\Delta U^*)^{(n+1)}$. For this reason it is convenient to run them asynchronously with different time-steps.

3.1.1.3 Adaptivity in Space-Discretization

The adaptivity in space is done through mesh refinement and unrefinement. These are performed where a-posteriori error estimates are above (resp. below) a certain error tolerance.

A-posteriori error estimates are obtained using the Zienkiewicz-Zhu method, which has been described in Chapter 2. This method has been intensively used mainly by structural engineers, but, as has been moticed, it is independent from the underlying differential problem. Hence it can be applied in general also to other problems, e.g. to time-dependent fluid flows problems.

In this example the estimates on water flow approximation error were performed separately for velocity and for elevation. The error estimates are then used for deciding local mesh refinement (i.e. element subdivision) or coarsening. In the problem at hand, the process of mesh adaptation turns out to be effective for improving the approximation of water elevation and less useful for improving the approximation of water velocity (for which time-discretization error plays a greater role).

The mesh adaptation algorithm uses a-posteriori error estimates for obtaining an *equilibrated* mesh (i.e. each element gives the same contribution to the error) with the total error below a certain predetermined tolerance. It is a slight modification of one given in [100] for first balancing the mesh and then lowering the total error. Mesh adaptation occurs when the error in water elevation or the error in water velocity requires a mesh-change.

Mesh generation and refinement was done performing Delaunay triangulation. This can be accomplished today reliably and fast thanks to existing algorithms that can be found on the Internet (see Chapter 4).

3.1.1.4 Adaptivity in Time-Discretization

To obtain error estimates for $(\Delta U^*)^{(n+1)}$, since for discretizing U^* we have used a Taylor series expansion, it seems natural to calculate the discretization error using the remainder in the Lagrange form; it holds that:

$$(\Delta \mathbf{U}^*)^{(n+1)} = \Delta t \left(\frac{\partial \mathbf{U}^*}{\partial t}\right)^{(n)} + \frac{\Delta t^2}{2}\left(\frac{\partial^2 \mathbf{U}^*}{\partial t^2}\right)^{(n)} + \frac{\Delta t^3}{6}\left(\frac{\partial^3 \mathbf{U}^*}{\partial t^3}\right)^{(\xi)} \tag{3.13}$$

Now, taking into account (3.6) we obtain the expression for the third derivative of U^* :

$$\left(\frac{\partial^3 \mathbf{U}^*}{\partial t^3}\right)^{(\xi)} = \frac{\partial}{\partial t}\left(\frac{\partial^2 \mathbf{U}^*}{\partial t^2}\right)^{(\xi)} = \frac{\partial}{\partial t}(\mathbf{GR}_s)^{(\xi)} \tag{3.14}$$

where ξ is a point in $[t_n \quad t_{n+1}]$.

Then, after having determined the range $[U_{MIN}, U_{MAX}]$ of possible values of $U^{(\xi)}$ (e.g. by looking previous samples of the velocities or argued a-priori by the physical knowledge of the poblem into consideration), it can be taken the one that produces the maximum value for the discretized version of (3.14) :

$$\arg \sup_{\hat{U}\in[U_{MIN},U_{MAX}]} \left\{\frac{(\mathbf{GR}_s)(\hat{U}) - (\mathbf{GR}_s)(U)^{(n-1)}}{2\Delta t}\right\} = \arg \sup_{\hat{U}\in[U_{MIN},U_{MAX}]} \{(\mathbf{GR}_s)(\hat{U})\} \tag{3.15}$$

The maximum value could be computed by putting equal to zero the first derivatives of the expression between brackets, but this would require solving a nonlinear system of equations; actually we are interested in the maximum over a little interval of values for U and this can be done in a simple way looking at the behaviour of the relatively smooth function $(\mathbf{GR}_s)(\hat{U})$. The efficiency of the estimator depends much on the choice of U_{MIN} and U_{MAX}; a too small interval would bring to under-estimation of the error, while a too large interval would bring to over-estimation of the error. The range of values indicated by U_{MIN} and U_{MAX} would however remain in a certain measure close to the optimal one (i.e. the range which would give the correct error estimate) since an unstable growth of the acceleration due to a too large Δt (i.e. under-estimation of the error) would bring to a raise in the range (i.e. to a potential increase of the error estimates and consequently a reduction of the Δt), while an ever stable velocity value would bring to a reduction in the range (i.e. to a potential decrease of the error estimates and consequently an increase of the Δt). Hence the adaptive scheme is at least stable.

The time-step can be adaptively modified with the simple rule: $\Delta t_{n+1} = q * \Delta t_n$ where $q \leq (\frac{\epsilon}{error_{estimated}})^{\frac{1}{p}}$ and ϵ is the predefined error tolerance, $error_{estimated}$ is the a-posteriori time-discretization error estimates derived above and here $p = 2$ because the Taylor series expansion was of order two (see [67]).

We have obtained an expression for the so-called 'local error' (i.e. the error in one step). This serves us for adaptively changing the time-step to maintain the local error below a predetermined tolerance. For purposes that will be explained on section 5, we need also an estimate of the so-called 'global error', that is the total approximation error in the mesh nodes due to time discretization. For this purpose we must "transport to current time all previous local error and add them up" (see [67] pp.160-161). In our situation this means finding for each mesh node k two constants C_k and L_k , namely vector $L = [L_k]$ is an upper bound for the norm of $\mathbf{G} = \dfrac{\partial \mathbf{R}_s}{\partial \mathbf{U}^*}$ and can be obtained by evaluating at for each node the maximum velocity observed; $C = [C_k]$ is the vector of constants where $|e_k^{(n)}| \leq C_k * \left(h^{(n-1)}\right)^{p+1}$ is satisfied at each mesh node, with $e_k^{(n)}$ the local error estimate obtained before. Then the following expression gives the global error for the node k:

$$|E_k^{(n)}| \leq h^p \frac{C_k}{L_k} \left(\exp(L_k(U_k^{(n)} - U_k^{(0)})) - 1\right)$$

To obtain Error estimates for $(\Delta U^{**})^{(n+1)}$, let us consider eq. (3.12); a simple local error estimate for this series expansion is the following:

$$e_{u_i^{**}}^{(n+1)} = \frac{\theta_1^2}{2}[\Delta u_i^{*(n+1)} + \Delta u_i^{**(n+1)} - \Delta u_i^{*(n)} - \Delta u_i^{**(n)}] \qquad (i = 1, 2)$$

$$e_{\eta^{**}}^{(n+1)} = \frac{\theta_2^2}{2}[\Delta \eta^{**(n+1)} - \Delta \eta^{**(n)}]$$

3.1.2 Space-Time Finite-Elements Examples

In a series of papers, Eriksson e Johnson have formulated adaptive space-time finite elements discretizations for parabolic problems [53] [54] [56] [57] [58].

Johnson and collegues have also applied these techniques to conservations laws and fluid dynamics problems in general [82] [81] [68].

The methods presented in these papers do use *explicit* a-posteriori error estimators (cfr. Section 2.2.1), which are residual based and hence they mix together the error contribution in space and in time. We will see in Section 3.3 that this does not bring to an efficient adaptation.

Moreover, they use equal time-steps for all elements, instead e.g. in [75] it is shown the convenience of building an unstructured mesh also in time (i.e. variable step-sizes in time for the elements, as it is commonly done in space). In this thesis it is presented an a-posteriori estimation scheme which refines locally and directionally in the space-time domain, hence it is suited for building an unstructured mesh also in time.

3.2 Space and Time Errors Interaction

An important issue in adaptivity is to obtain an efficient sequence of discretizations, i.e. the total number of degrees of freedom must be kept as low as possible. In the literature, a-posteriori error estimation and mesh adaptation are made independently for space and for time. Actually, when using an adaptative algorithm for solving a time-dependent problem, one must consider the possibility that the discretization error in time may corrupt the a-posteriori error estimates in space and vice-versa. In fact, $U = u + e_s + e_t$, where e_s is the error due to space discretization and e_t is the error due to time discretization, and the a-posteriori error estimators work on U .

For example, the effect of a too high time-step is that the error in space is usually under-estimated. Let us consider again the example of shallow waters equations of Section 3.1.1 [36]: in this case, the numerical experiments indicate that two adaptive-in-space simulations with different Δt, after three hours of simulation, yield these results:

Δt	number of mesh nodes	$\parallel \sigma_{e_s} \parallel$ for water elevation
10	1465	45
30	1736	38

From a-posteriori error estimates (in space only) the simulation with $\Delta t = 30sec$ appears to be more accurate than the other, but Figure 18 shows that actually the adaptive-in-space simulation with $\Delta t = 30secs$ is less accurate than that with $\Delta t = 10secs$. Hence, making a-posteriori error estimates only in space, for time dependent problems, does not produce reliable estimates even of space-discretization error alone. An intuitive explanation of this fact, quite commonly agreed, is that with increasing time-steps the spatial pattern of water elevation becomes smoother; hence the a-posteriori error estimator 'see' a simpler function to approximate than the true one, so they believe the approximation error is lower than it is truly. We could say that, due to time-discretization error, finer-scale phenomena are lost. The addition of a-posteriori error estimates of the time-discretization error can rescue this situation.

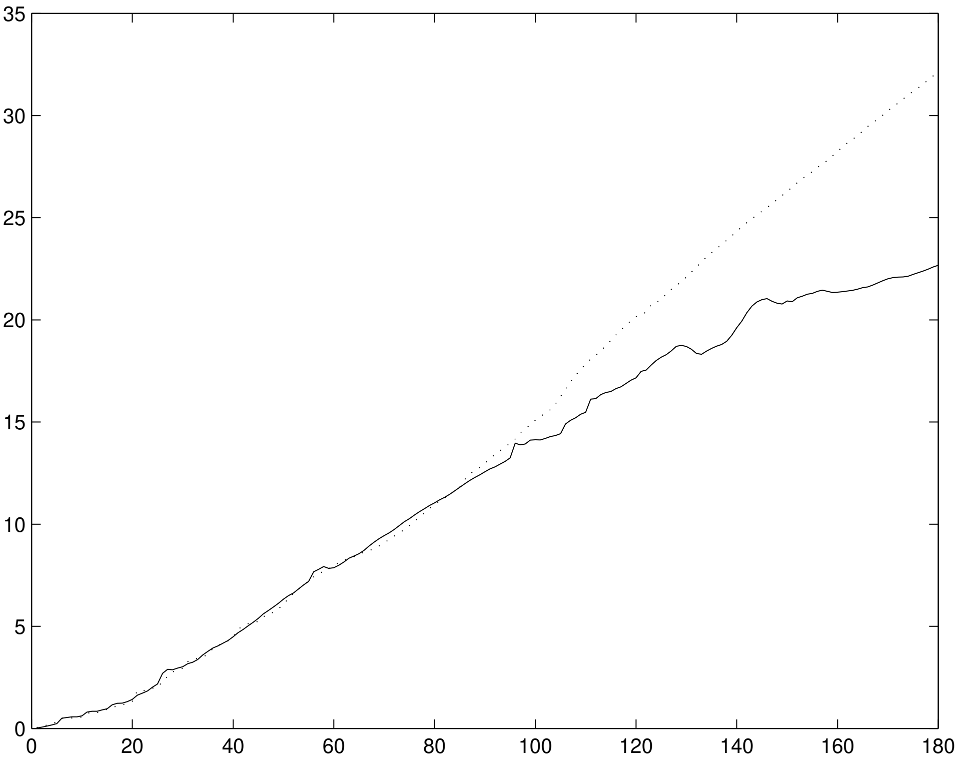

Figure 18: total approximation error with different constant time-steps: 10sec (continuous line) 30sec (dotted line), versus elapsed Time (minutes)

Now, it may be difficult to determine a-priori if, with the error-tolerances specified, there will be relevant interaction between space and time adaptations. Let us investigate this issue using the ZZ-estimator. For simplicity we wiil consider its 1987 version, in which $\bar{\sigma}_i^*$ values are found such that the Galerkin-orthogonality condition is satisfied:

$$\int_\Omega N^T\left(\sigma_i^* - \hat{\sigma}_i\right) \mathrm{d}\Omega = 0 \qquad (i = 1, 2, ..., n_d)$$

Let the Zienkiewicz-Zhu criterion be applied to U to obtain a-posteriori error estimates as showed in Chapter 2:

$$\int_\Omega N^T\left(\sigma_U^* - \hat{\sigma}_U\right) \mathrm{d}\Omega = 0$$

where now $\sigma_U = \sigma_{u+e_s} + \sigma_{e_t}$; hence

$$\int_\Omega N^T(\sigma_{u+e_s}^* + \sigma_{e_t}^* - \hat{\sigma}_{u+e_s} - \hat{\sigma}_{e_t})\mathrm{d}\Omega = \int_\Omega N^T(\sigma_{u+e_s}^* - \hat{\sigma}_{u+e_s})\mathrm{d}\Omega + \int_\Omega N^T(\sigma_{e_t}^* - \hat{\sigma}_{e_t})\mathrm{d}\Omega = 0$$

that in matrix form, becomes $M\sigma_{u+e_s}^* = Q\hat{\sigma}_{u+e_s} + (Q\hat{\sigma}_{e_t} - M\sigma_{e_t}^*)$ .Note that this expression is equivalent to that calculated in Chapter 2:

$$M\sigma_U^* = Q\hat{\sigma}_U$$

If we choose $\sigma^*_{e_t}$ such that $M\sigma^*_{e_t} = Q\hat{\sigma}_{e_t}$, we obtain the a-posteriori estimate of the discretization error in space, free from perturbations produced by the discretization error in time:

$$\| \sigma^*_{u+e_s} - \hat{\sigma}_{u+e_s} \| = \| (\sigma^*_U - \sigma^*_{e_t}) - (\hat{\sigma}_U - \hat{\sigma}_{e_t}) \|$$

The same can be formulated, with the necessary modifications, for eliminating the influence of discretization error in space on a-posteriori error estimates of discretization error in time.

Let us see now what happens to the errors interaction when adaptive space-time elements are used. In the semi-discretization algorithm [36], as previously shown, the a-posteriori error estimates in space are disturbed by an extra-term depending on the approximation error in time e_t:

$$(Q\hat{\sigma}_{e_t} - M\sigma^*_{e_t})$$

Since e_t can be only estimated, the correction to the estimate in space cannot be exact. With adaptive space-time finite-elements, instead, a single mechanism can be used to estimate both the discretization errors in space and in time. Hence the interactions between the two estimates are reasonably reduced or even eliminated.

3.3 Directionality in Space-Time Error Estimation

In the presence of phenomena which propagate in the space-time domain, it is better to know the approximation error which is encountered on each direction (in a space-time reference coordinate system), to achieve an efficient space-time mesh adaptation. This is even more important when the solution have strong gradients which propagate following a precise direction (e.g. shock-waves).

When the solution possesses strong gradients towards distinct space-time directions, the approximation error is higher in the same directions.

This directionality in the evolution of the solution can be advantageously exploited by a-posteriori estimators of the ZZ-type, since the estimates are derived component-by-component (with respect to the reference coordinate system). The a-posteriori error element indicators can be e.g. computed separately for space and time components:

$$\|\epsilon_{space}\|^2_{\Omega\times[0,T]} = \int_\Omega \epsilon^T_{space}\epsilon_{space}\,\mathrm{d}\Omega \times [0,T] = \sum_{k=1}^{m} \left[\int_{e_k} \epsilon^T_{space}\epsilon_{space}\,\mathrm{d}e_k \right] = \sum_{k=1}^{m} \|\epsilon_{space}\|^2_{e_k}$$

$$\|\epsilon_{time}\|^2_{\Omega\times[0,T]} = \int_\Omega \epsilon^T_{time}\epsilon_{time}\,\mathrm{d}\Omega \times [0,T] = \sum_{k=1}^{m} \left[\int_{e_k} \epsilon^T_{time}\epsilon_{time}\,\mathrm{d}e_k \right] = \sum_{k=1}^{m} \|\epsilon_{time}\|^2_{e_k}$$

where e_k is the $k-th$ space-time element.

Now that the components of the error with respect to a reference coordinate system are known, it can be derived the usual magnitude of the error in each element (measured e.g. in the L_2 norm or in the energy norm) but also the direction in which the error is maximum, i.e. the *error direction.*

This property is not owned naturally by a-posteriori estimators based on the residuals, since the residual represents both the approximation errors in space and time, mixed together. In a situation in which the mesh in space is adequate and it is necessary to refine it in time, they would refine also the mesh in space, loosing therefore the efficiency of the adaptation. This is certainly true for *explicit* estimators, while *implicit* estimators use the residuals only as an error model and then they solve an auxiliary problem to approximate the error model (cfr. Chapter 2). Looking at the approximate solution of the error model equation, they can actually infer also the direction in which the error is higher. For this reason also the novel estimator (i.e. the method of *local extensions update* presented in this thesis on Chapter 2) have this capability.

We have seen the advantages of having a unique estimation mechanism for space and time discretizations. Now, if the estimator is also capable of distinguishing between the space and the time components of the error, it is possible to obtain an efficient mesh adaptation. Let us assume the *effectiveness* of the ZZ estimator. This means that its *effectivity index*, cfr. the beginning of Chapter 2, is reasonably close to 1. If this is true not only for the overall error, but also with respect to each component of the true error, we say that it is *efficient*. With space-time elements, the ZZ-method can estimate both e_t and e_s at the same time and distinguish the two components, and hence to be also *efficient*.

The application of the ZZ-method to a space-time finite element discretization (cfr. Section 4.2) of the shallow waters model previously presented in this Chapter (Section 3.1.1) led to the following results [39]: in the following table are showed the values of the mean error estimates after one hour of simulation, on the same space-time patch of elements, when different constant time-steps were applied.

Δ t (secs)	space-error estimate	time-error estimate
1	0.4547	0.0002
5	0.4550	0.0011
10	0.4689	0.0020

It can be noticed that time-error estimates increase substantially with Δt while space-error estimates remain quite constant.

PART II

Mesh Adaptation

Chapter 4

4 Mesh Refinement/Un-Refinement

In the first part of this Chapter, some techniques of mesh refinement/un-refinement for space and space-time finite elements discretizations are presented (for a more exhaustive presentation, see e.g. [71] [27]). The main properties, from the computational point of view, of a meshing technique are put in evidence.

In the secod part of the Chapter, a novel recursive discretization scheme and its topological and computational characteristics will be presented.

4.1 Space Domain Discretization

The adaptivity in space is done through local mesh refinement and un-refinement. These are performed where a-posteriori error estimates are above (resp. below) a certain error tolerance.

Let us see an example [36]. Here the mesh adaptation algorithm uses a-posteriori error estimates for obtaining a balanced mesh (i.e. each element gives the same contribution to the error) with the total error below a certain predetermined tolerance. It is a slight modification of one given in [100] which first balances the mesh and then lowers the total error. Since in this problem the same mesh is used to discretize two equations solved in different variables (water elevation and water velocity), the mesh should result satisfactory for the approximation errors in both variables. For this reason the mesh refinement is made if required by only one of the two variables, while mesh un-refinement is made only if permitted by both variables.

In [36] mesh generation and refinement was done performing Delaunay triangulation. This can be accomplished today reliably and fast thanks to existing algorithms that can be found also on the Internet [93].

The unrefinement step is a little more delicate. In general, if the refinement has been made according e.g. to a tree-hierarchy, then an un-refinement step can be made by restoring the previous layer of elements or by modifying easily the hierarchy. If some form of hierarchy is not present, the unrefinement must in general comprehend some form of local remeshing. In [36] in an un-refinement step the mesh nodes are eliminated individually where all adjacent triangles have error indicators below the desired tolerance, creating in this way

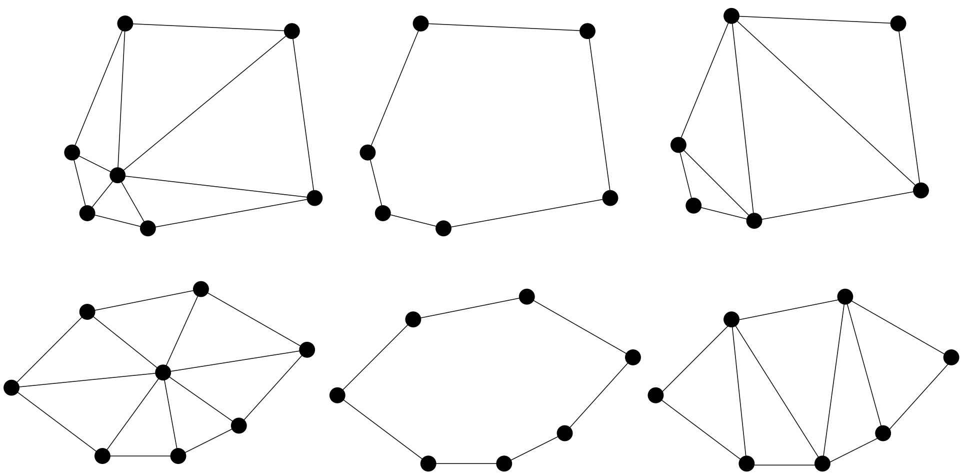

Figure 19: two un-refinement examples

'polygons' that are successively triangulated (again Delaunay) possibly without adding interior nodes.

This procedure seems to have nice properties:

- the number of triangles eliminated is fixed (two every node cancelled); in this way we can control the areas of triangles also during unrefinement and this is a good property for an adaptive algorithm;
- the unrefinement does not degrade the mesh quality, and may enhance it locally (see Figure 19)

In the literature exist many approaches to build a mesh, and the refinement/un-refinement techniques are in general dedicated to the specific topology of the mesh.

For adaptive finite element solvers, two are the main aspects which can be strongly influenced by the topology and the level of structure owned by the mesh formulation. The first one is the quantity of labour which is necessary to do the refinement/un-refinement operations and to implement the a-posteriori error estimator (especially if it is based on extrapolation techniques, like that of Chapter 2). This depends primarily on the data-structures that it must be worked on and the quantity of information contained in them. The second important aspect is the cost required to get the solution of the resulting linear system of equations.

For the first aspect, a hierarchic formulation of the mesh is helpful; in general it can be a *tree*: there is a coarse initial mesh which is the *root* and successive refinements are subdivided in *levels* of the tree. An example of hierarchic meshes can be given by the *Hierarchical Bases* [15], in which it is imposed locally a pre-dermined topology to the refinement. This local constraint turns out to be useful also for the analysis of the resulting discretization: the local shape regularity property substitutes the usual global counterpart required for demonstrations (the advantage is that some properties can be demonstrated for more general meshes than usual).

in this Thesis it is proposed (cfr. Section 4.3) a hierarchic approach to the construction of a mesh which does not impose local constraints to the topology. It results a completely un-structured mesh which possesses a tree-like hierarchy.

For the second aspect, the topology of the mesh can influence e.g. the condition number of the system matrix (thus also on the speed of convergence to the solution if iterative methods are used to solve the system) or influence the production of fill-in during the factorization of the matrix if direct methods are used.

For example, the *Hierarchical Bases* [15] give a much lower condition number than the usual *Nodal Bases* (which are the most used in finite elements); hence they permit a faster convergence for iterative methods [102].

The method proposed in Section 4.3 have shown to produce low-fill in in both problems of solving completely the system of equations and of eliminating only the nodes which are internal to an initial coarse mesh (see Section 4.3 and 5.1.3.3).

4.2 Space-Time Domain Discretization

The underlying principles which lead to the construction of an adaptive space-time finite-elements mesh are the same as that presented in the previous Section, but here one dimension corresponds to the time-variable. We have seen on Chapter 3 that the directional a-posteriori error etimates must be completed with an appropriate directional mesh adaptation.

Let us see an example of space-time mesh. The space-time *element* here considered is a prism with triangular base (belonging to the space domain) and height orthogonal to the base (belonging to the time domain). The space-time *slab* here considered consists of one layer of elements, hence its depth in time is the same depth of each element inside the slab.

Let us consider figure 20. If we use a finite element discretization which is continuous also in time, nodes $P4, P5, P6$ must be given data (i.e. initial conditions for the slab), while nodes $P1, P2, P3$ are unknowns. This space-time element is a linear prismatic element whose shape functions are defined as the volume-ratio of the two prisms indicated in Figure 20:

$$N_1(x, y, t) = \frac{vol(Prism(A, B, P5, P6))}{vol(Prism(P4, P1, P5, P6))}$$

where

$$vol(Prism(P4, P1, P5, P6)) = \frac{1}{6} \begin{vmatrix} 0 & x_5 - x_4 & x_6 - x_4 \\ 0 & y_5 - y_4 & y_6 - y_4 \\ t_1 - t_4 & 0 & 0 \end{vmatrix}$$

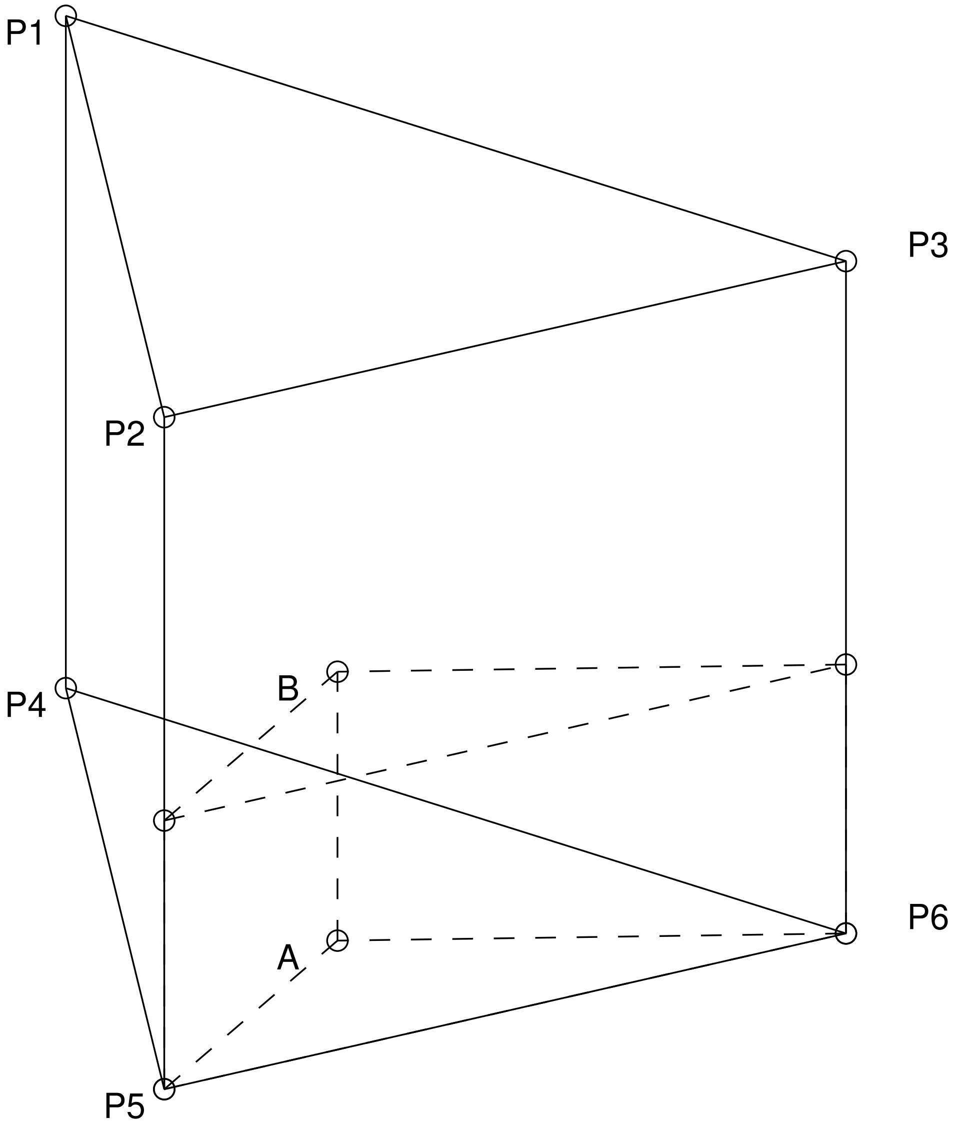

Figure 20: Prismatic Space-Time Finite Element

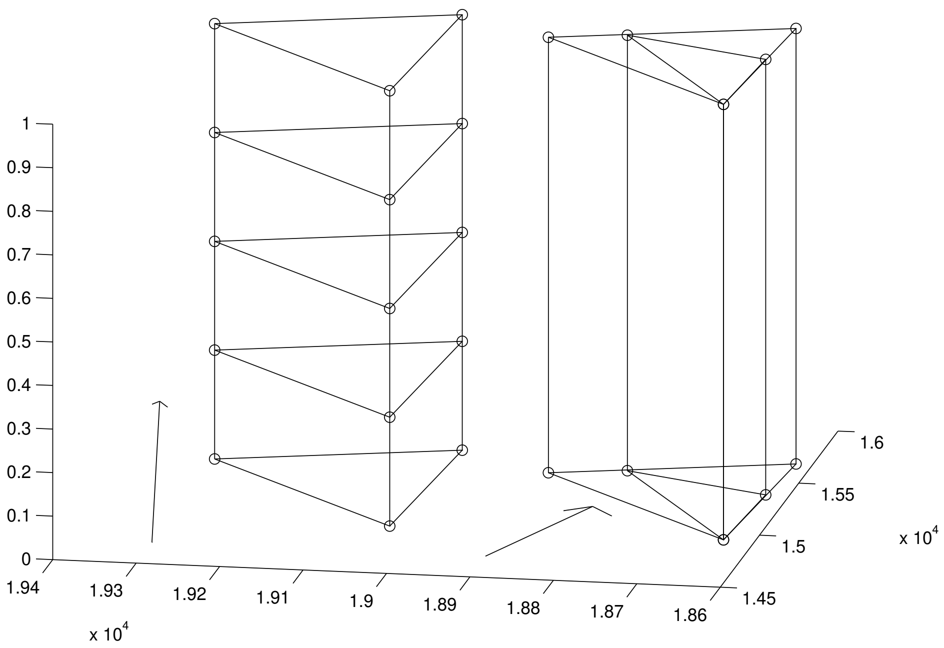

Figure 21: Directional Refinement examples: error direction orhogonal to the space domain (left) and error direction orthogonal to the time domain (right)

and similarly for $N_2(x, y, t)$ and $N_3(x, y, t)$.

Let us consider the mesh refinement/un-refinement problem. Here this has been done taking into account the *error direction* information made available by the directional a-posteriori error estimates presented in Chapter 3. this information leads naturally to a directional mesh adaptation, e.g. the longest element dimension of the refined elements can be enforced to be orthogonal to the *error direction*.

The topic of mesh orientation has been already considered in the literature and it is widely recognized that in many problems it is important that, other than equidistribution of the error among elements, the elements are properly oriented. Various motivations led researchers to develop mesh orientation methods; among others, Hansbo [68] uses structured meshes of triangles obtained by subdividing quadrilaterals with one of the two possible diagonals and gives orientation to the mesh by switching the diagonals individually as needed. Masud and Hughes [87] use space-time finite elements and perform mesh orientation for time-dependent problems with moving domains. Zienkiewicz and Wu [110] develop an automatic mesh generation procedure that creates a structured mesh of elongated elements where indicated by the curvature of the solution and an unstructured mesh elsewhere. Here we approach the construction of an oriented space-time mesh mainly through directional mesh refinement-refinement based on a-posteriori error estimates.

In principle, mesh refinement in the direction of the time-variable can be done non-uniformly in space, creating space-time elements patches of the type shown in Figure

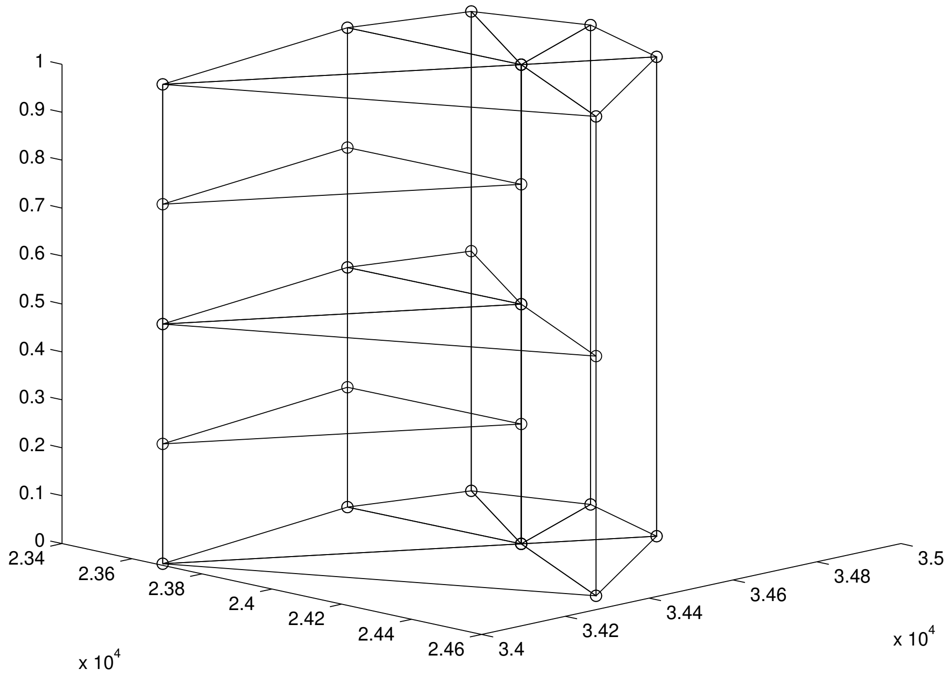

Figure 22: Space-Time Elements Patch

22.

From the figure it can be seen that this kind of mesh suffers of the fact that not all elements can be linear and in principle requires the definition of "transition elements" [73]. If are used tetrahedra instead of prisms, the problem disappears.

This type of mesh is composed by *slabs* which consist of more than one layer of elements in the time-direction. This fact means that also the discretization in time is made simultaneously for some intermediate steps: the error estimates in time result in this case more globally determined and their sharpness can be bigger since with a deep *slab* they can be made simultaneously in a quite long time-interval and thus can be partly avoided the need of transporting previous discretization errors to the present [67], that is typical of semi-discretization methods. The depth of the slab increase substantially the number of equations that must be solved simultaneously in the system. The use of the techniques presented on the third Part of this Thesis for this kind of problem could be useful to reduce the computational cost, and is under development.

Note that in this way the approach to adaptivity is to use space-time finite elements and the adaptation in time is made locally with respect to space (i.e. in certain space regions the simulation advances with higher time-steps than in other regions) and, if it is convenient, also in an unstructured way (as usually happens for adaptivity in space). Since, usually, physical phenomena evolve at different speeds in different regions in space, this approach brings to a more efficient discretization.

Space-time meshes can thus be topologically structured in time or completely unstructured. For one-dimensional problems in space (hence bi-dimensional in space-time) the corresponding elements used are usually quadrilaterals and triangles. Quadrilaterals can be refined and oriented in time so that edges parallel to the time axis be orthogonal to the *error direction*. Triangles can be refined inserting an edge that is orthogonal to the *error direction*.

For bi-dimensional problems in space (i.e. tri-dimensional in space-time) can be used prismatic elements or tetrahedra. With prismatic elements the error component parallel to the time axis is used independently to determine the height of each prism and its orientation, and the *error direction* in space is used to refine/un-refine and possibly orient the prism basis (like in Figure 21). Tetrahedral elements are refined inserting a face that is orthogonal to the *error direction*.

Note that in this way the resulting elements are elongated and oriented in a direction orthogonal to the *error direction*. In this way, final mesh density is higher in the error direction and lower in its orthogonal direction(s). This situation contributes to keep low at the same time the approximation error and the number of elements.

4.3 A Recursive Unstructured-Mesh Formulation

The aim and the general philosophy of this new meshing technique are described heretoafter. Since mesh generation is a well-established discipline in computational geometry and several algorithms are available today which produce good discretizations of bi- and tri-dimensional domains, the general approach has been that of using them as building blocks of a general technique which possesses additional properties that are useful for one or more specific classes of problems.

The classes of problems here referred are that of adaptivity and of arbitrarily varying coefficients [38] in finite elements applications. Hence, other than fulfillment of standard *aspect ratio* (e.g. *minimum angle* conditions [5] if it is a mesh of triangles), the resulting mesh should be able to follow all the material interfaces and coefficients variations patterns, and to approximate locally the solution to the desired level. This implies that the construction of the discretization must be possible with enough local-flexibility (i.e. the mesh must be locally un-structured).

Moreover, the resulting mesh must satisfy the two main computational requirements indicated in Section 4.1, i.e. to be simple to handle by software algorithms and to permit a fast resolution of the corresponding algebraic problem (i.e. the solution of the system of equations which arises from the discretization of the PDE problem on this mesh using finite elements) and do not require huge amount of memory for problem solution.

For simplicity, in the following we take into account only triangular elements. The extension to rectangular elements or to three dimensional ones (tetrahedra and hexahedra) will be considered in the future as an extension of this technique. The recursive mesh generation here presented uses as a building-block a general mesh generation algorithm existing in the literature, whose development is due to J.Ruppert [92] and numerical implementation to J.Schewchuk [93]. It produces Delaunay triangulations of a general polygonal domain. See the cited references for theoretical proofs about the quality of resulting triangles (i.e. a bound on the minimal internal angle of the triangles) and the efficiency of the triangulation (i.e. the size of the mesh is at most a constant factor of the best possible). This algorithm is used here e.g. to produce automatically the *initial mesh* and then to generate the subsequent recursive layers of nodes, as described in the following sub-section. The hierarchy of nodes obtained from this recursive technique is then used for the stiffness matrix assembly and for the subsequent solution of the resulting linear system of equations, with the advantages delineated in Section 4.3.2 .

4.3.1 Recursive Refinement

For the classes of problems here considered, the recursive refinement can be done a-priori, based on the knowledge of the coefficients variations, or a-posteriori, based on a-posteriori error estimates. Let us present the a-priori algorithm; its derivation for adaptivity will be presented in Section 4.3.3 . The recursive algorithm is the following:

1. **creation of the initial triangulation**: this is automatically generated from a description of the (internal) and external boundaries and creates the first layer of elements in the triangulation. This can be done by applying a general triangulation algorithm to the overall domain or to a decomposition of it in a set of connected polygons (a sort of the well-known *sub-structures* used in finite elements discretizations). Successively, an optional mesh-smoothing step is performed to place the nodes on interfaces that are closer than a pre-defined distance $LMIN$ along the edges.

2. **nodes insertion step**: on all new added edges, nodes are added where the coefficient variation along the edge, from the neighboring existing/added nodes, is bigger than a prescribed threshold (ΔC) and the distance from the neighboring existing/added nodes is bigger than a prescribed minimum value ($LMIN$).

3. **local refinement step**: each triangle which has new nodes added on its edges is again Delaunay triangulated (this can be done with or without adding Steiner points); this creates new edges in the triangulation and a new layer of triangles.

4. If the *local refinement step* have added new edges in the triangulaton, call again (recursively) this algorithm from step 2;

5. End

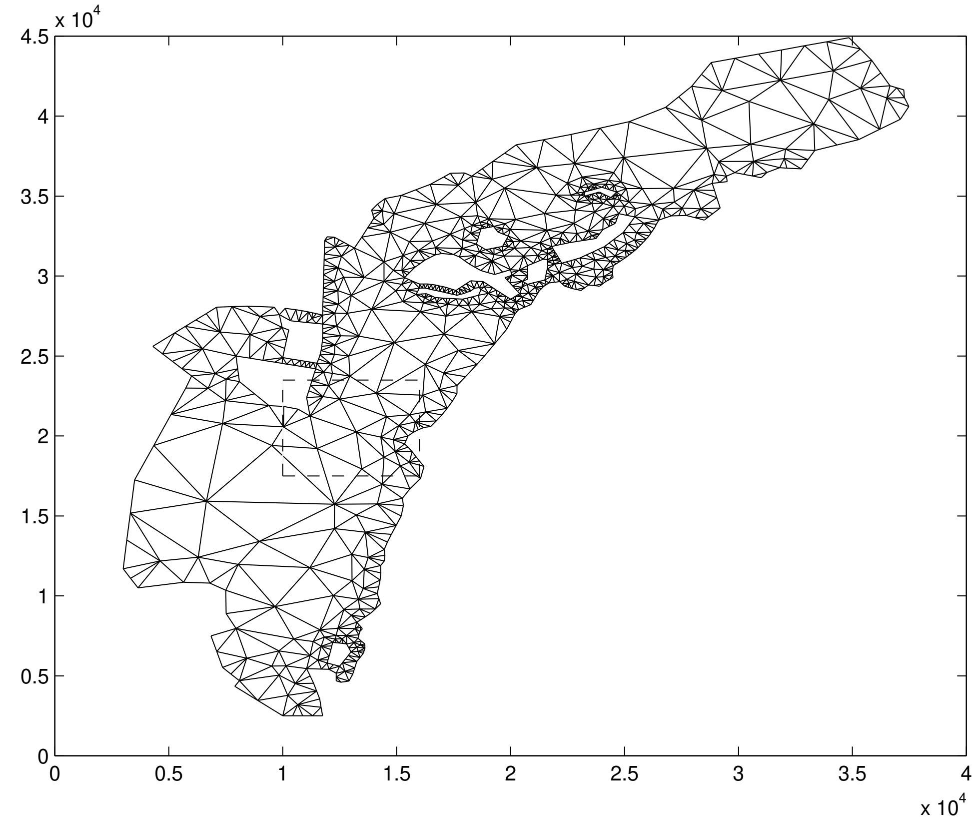

Figure 23: the *initial mesh* - (the dashed box is the zone of figg. 24 and 25)

Let us briefly analyze this algorithm. As a consequence of the *nodes insertion step*, in the case of steep variations of the coefficient (due e.g. to an interface between different materials) the nodes are added exactly on the interfaces.

As a consequence of the *local refienement step*, the final mesh resulting from the algorithm has triangles with predetermined quality, because each recursive refinement of the triangulation is obtained from a quality Delaunay triangulation algorithm applied to a polygonal subdomain.

The recursion mechanism in the algorithm stops when in no edge of the triangulation there is a variation of the parameter values greater than the user-specified threshold ΔC at a distance bigger than $LMIN$ from a mesh node. This fact guarantees that the parameter is constant, up to the predefined variation ΔC, on all the three edges of each triangle in the resulting mesh. This is sufficient in most applications to assure that it be almost constant also inside the triangle and that the contours of the interfaces are reasonably tracked by the edges of the triangulation. If required, however, it is easy to introduce an additional subdivision mechanism based on variations of the coefficient inside the triangle: the subdivision creates one or more additional edges upon which the standard algorithm can be applied. Moreover, the position of the nodes can be adjusted to optimize the tracking of the interfaces with the edges. This adjustment actually lowers the number of

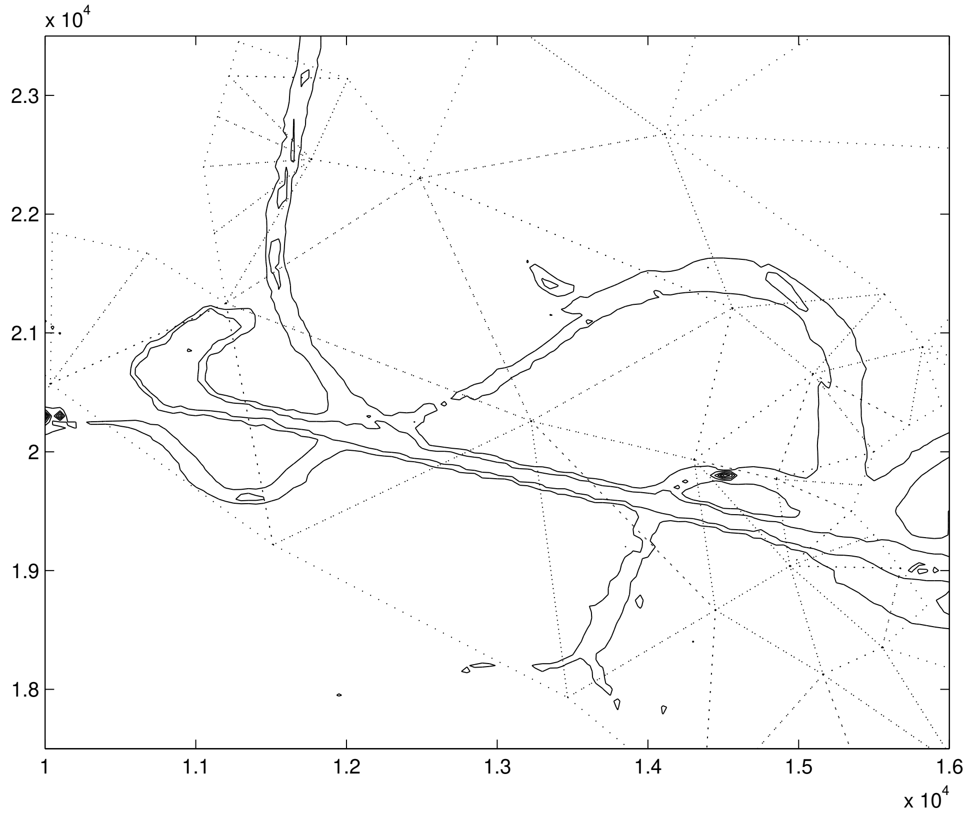

Figure 24: the *initial mesh* (dotted) - (continuous lines are water-depth level-curves)

nodes generated by the algorithm.

The parameters ΔC and $LMIN$ play an important role: the former defines the accuracy on the discretization of the PDE parameter, the latter defines the minimum spatial scale for the coefficient variations that will be resolved by the solution. The total number of degrees-of-freedom may be quite sensitive to the value assumed by these parameters. The parameter $LMIN$ determines also the minimum length of mesh edges (hence it controls also the mesh/size parameter h, cfr. Chapter 1), that cannot be lower than a fraction of it.

In Figg. 23,24,25 it is shown an application of this algorithm to a problem of shallow waters in which the varying PDE coefficient corresponds to the depth of the water in the stationary state and varies in a highly irregular way. In Figure 25, the triangulation was made without adding Steiner points; this was done for better recognizing the nodes added on the edges of the triangles, but this is not the operative choice since without Steiner points added there are in this case too much bad-shaped triangles (i.e. with too small internal angles).

It is important to note that this recursive algorithm generates a discretization that can be viewed as a hierarchy of nested finer discretizations that by construction identify groups of nodes which have the property to be *internal* to the triangles of the preceding higher

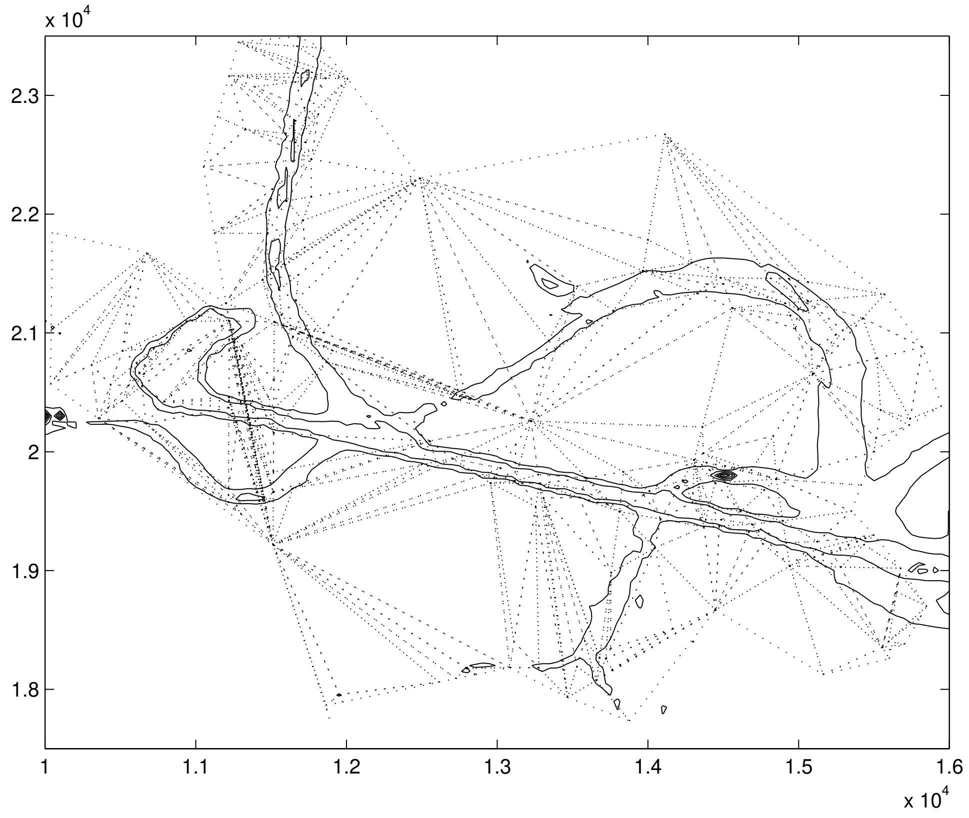

Figure 25: recursively refined triangulation after seven levels of recursion (dotted), without Steiner points added

layer. This hierarchy and its properties come out to be a key element for simplifying the necessary operations for solving the linear system of equations.

This algorithm generates some geometric entities that can now be defined:

- the triangles belonging to the mesh generated at step 1 and the mesh itself will be called in the sequel the *initial triangles* (other than *layer-0 of triangles*) and the *initial mesh*, respectively;
- the new subdivision created after the refinement step is made to the layer-$(n-1)$ of triangles, determines the *layer-n of triangles*;
- the new edges created by the initial triangulation (i.e. layer-n of triangles, with $n = 0$) or by the local refinement step made to the triangles of the layer-$(n-1)$ of triangles, will be called *level-n of edges*;
- the nodes placed on the edges of the layer-$(n-1)$ of triangles, constitutes the (*level-n of nodes*. The level-1 of nodes plus initial nodes constitutes the *interface nodes*. Each group of nodes created at the level-n are by construction *internal* to a corresponding triangle of layer-$(n-2)$.

Thanks to this recursive discretization strategy, the assembly of the stiffness matrix is made in two steps:

1. all the submatrices corresponding to each set of nodes that are internal or lies on edges and vertices of an *initial triangle* are assembled in parallel;

2. the global stiffness matrix is assembled from the submatrices obtained at the previous step, after the equations corresponding to the internal nodes have been eliminated by Gaussian elimination. This global stiffness matrix is called also *homogenized matrix* and the corresponding mesh is called *homogenized mesh* [38].

Note that in this way the problem dimension is not given by the total number of equations, but is that of the homogenized mesh; this fact reduces considerably the amount of memory required to do the matrix assembly and to solve the corresponding system of equations with a direct method. Moreover, the sparsity of the homogenized matrix is almost the same of a stiffness matrix obtained by direct assembly on the mesh consisting of the *initial mesh* plus the level-1 of nodes and the first layer of triangles. Note that here the parallelism in doing mesh assembly and homogenization arise naturally, by construction and can be decided a-priori while creating the initial mesh.

4.3.2 Algebraic Advantages

It turns out that the hierarchy of nodes and elements generated by the recursive discretization is very useful for the solution of the resulting linear system of equations: all the operations that generally would be required to decide how to do the calculations (mainly unknowns reordering and elimination tree determination) and how to redistribute the calculations if more than one processor is available, here are automatically determined a-priori and automatically by construction.

Moreover, the problem is naturally dissected into several smaller-size problems that can be solved independently, hence also memory requirements can be easily kept low as needed and there is an high intrinsic parallelism.

The solution is obtained by computing the Schur's Complement of the submatrices which correspond to the nodes that are internal to the initial mesh, with respect to the global stiffness matrix K: Given $Kx = f$ expressed in the form:

$$\begin{pmatrix} A_K & B_K \\ C_K & T_K \end{pmatrix} * \begin{pmatrix} d_x \\ s_x \end{pmatrix} = \begin{pmatrix} d_f \\ s_f \end{pmatrix}$$

and eliminating variables d_x (corresponding to the nodes internal to the *initial triangles*) we obtain the homogenized (*reduced*) matrix R_K:

$$R_K = T_K - C_K A_K^{-1} B_K$$

This corresponds to a numerical homogenization of the discrete problem, i.e. fine-scale phenomena are condensed into the initial mesh. This procedure have various good mathematical properties, e.g. it is demonstrated in the literature [46] that this reduction step preserves the ellipticity of the operator. This procedure is also well known in general in the Finite Element literature; it is used e.g. in what is called *static condensation* [107].

The variables d_x are eliminated starting from the last level of corresponding nodes in the recursive refinement and, precisely, following backward the order created by the recursive discretization algorithm of section 3. This is a key point for the following analysis and for the overall performances of the homogenization algorithm and of the overall solution of the linear system. In the following discussion, for simplicity, we will write "node" even when we refer to the corresponding column and row of the stiffness matrix.

For the Schur's complement computation (i.e. matrix reduction) we use Gaussian elimination (e.g. in the form of Cholesky factorization since for the class of problems here taken as an example the stiffness matrix is symmetric and strictly positive definite). The attention will be focused on three major issues: minimization of the fill-in, computational kernel efficiency and parallelism. The term *Fill-in* means that usually when the Cholesky factorization is applied to a sparse matrix the resulting factor has more non-zero entries than the lower/upper triangular part of the original matrix. This effect can be minimized with a proper re-ordering of matrix columns and rows: $(PAP^T)(Px) = Pb$ where P is a *permutation matrix*. Various heuristic algorithms for determining P exist in the literature [52]; two of them are considered the more effective: the *minimum degree algorithm* [62] [3] and the *nested dissection algorithm* [70]. The first one acts locally and therefore it is not well suited for parallelization [52]. On the other side the nested dissection algorithm is global, hence better suited for obtaining a balanced subdivision of the problem. Moreover the two algorithms may succesfully cohoperate: minimum degree re-orders effectively the groups of nodes determined by nested dissection.

In the recursive discretization here presented the natural ordering obtained by construction determines the global dissection and shows to be good both globally and locally.

Globally, the fill-in remains confined within the nodes belonging to each *initial triangle* (i.e. internal to the triangle or on his edges or vertices), see Figg. 26 and 27:

THEOREM 4.1: *"During the Cholesky factorization of a matrix generated by the recursive triangulation algorithm of section 4.3.1, following backward the order in which nodes are created, no fill-in entries are produced at indices corresponding to nodes belonging to different initial triangles"*

Proof: it follows immediately considering the fact that nodes to be homogenized are internal to *initial triangles* (i.e. no mesh-edge connects directly two nodes internal to two different *initial triangles*) and remembering that fill-in entries can be produced only by non-zero entries of the outer-product of the column and row that are eliminated (i.e. by combinations of nodes that are directly connected to the node corresponding to eliminated column and row). ♢

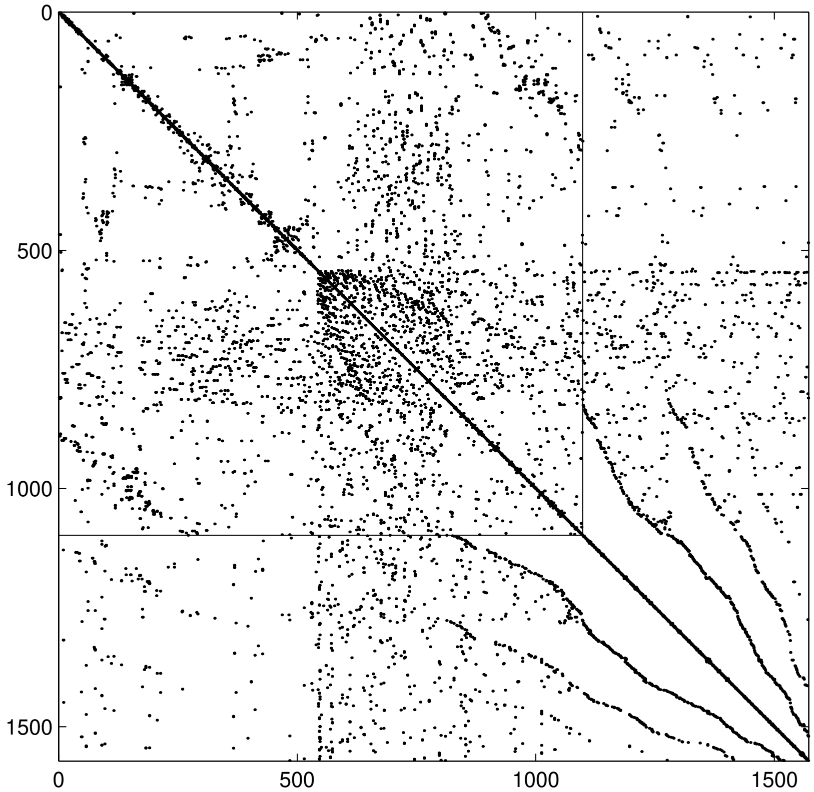

Figure 26: sparsity of the stiffness matrix for the recursively refined mesh (the square block inside represents *initial* plus *level-1* nodes, while the bottom-right part represents *internal nodes*

In Section 5.1.3.3 are presented some results of the low global fill-in produced by the natural ordering determined by this recursive meshing algorithm. Moreover, this sort of *dissection* is determined by construction (i.e. it does not require extra computations).

Locally, i.e. internally to each *initial triangle*, the nodes are grouped into sets that define triangles which contains the nodes of the subsequent levels. This creates a situation that ricursively resembles at each layer of triangles that of the preceding theorem. We can say that the fundamental property is that the $fill-in$ is constrained at each level by the maximum connectivity which is possible with the topology of the finite elements used.

The experimental results indicate that the natural ordering imposed by the recursive discretization algorithm creates no more fill-in than that created using permutations determined by the Minimum Degree Algorithm, the Approximate Minimum Degree algorithm, the Reverse Cuthill-McKee method and the Column Permutations method. See results on the following tables, where some results using different reordering methods for the problem of Fig. 25, are shown. In the first Table it was chosen $\Delta C = 5.0$, $LMIN = 50$ and various recursion levels were performed.

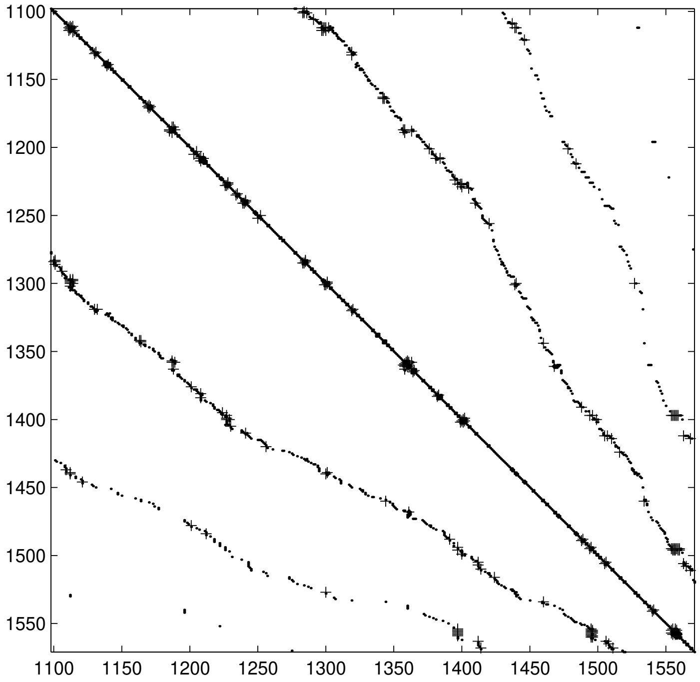

Figure 27: bottom-right part of the stiffness matrix (corresponding to *internal nodes*) with *fill-in* added (+)

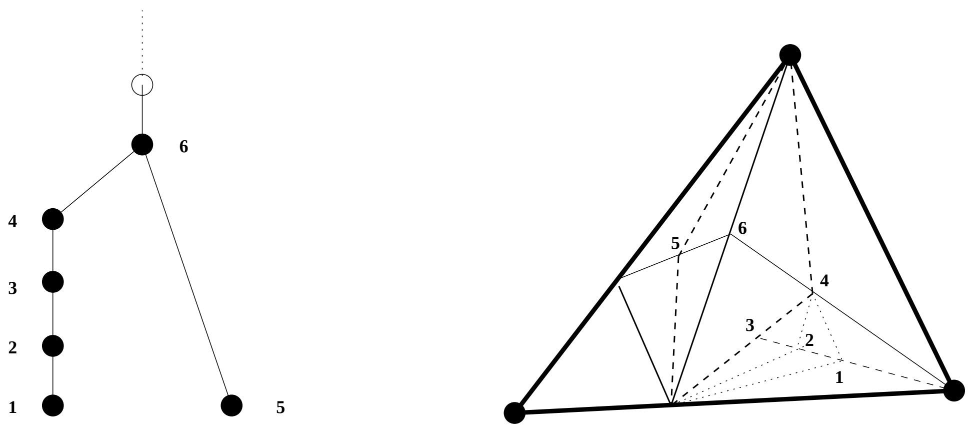

Figure 28: recursive refinement into an *initial triangle* (right): first layer of triangles (continuos line), second (dashed) and third (dotted). The corresponding *elimination tree* (left)

method	2 levels		5 levels		7 levels	
no reordering	850	10.9%	1790	18.2%	1964	18.7%
SYMMMD	772	9.9%	1628	16.6%	1938	18.4%
COLAMD	771	9.9%	1564	15.9%	1750	16.7%
columns permutation	770	9.9%	1584	16.1%	1814	17.3%
reverse Cuthill-McKee	836	10.7%	2568	26.1%	3842	36.6%
random permutation	842	10.8%	2670	27.2%	4484	42.7%

In the second Table it was chosen $\Delta C = 7.0$ and $LMIN = 50$.

method	3 levels		5 levels	
no reordering	488	7.3%	616	8.8%
SYMMMD	432	6.5%	580	8.3%
COLAMD	436	6.5%	570	8.1%
columns permutation	438	6.6%	574	8.2%
reverse Cuthill-McKee	500	7.5%	846	12.0%
random permutation	526	7.9%	950	13.5%

Another major issue is the efficiency of computational kernels; it is well known that for this purpose it is convenient to do dense matrix operations, e.g. using Level 3 BLAS subroutines. For this reason, the well-known multi-frontal method [85] re-organizes the overall Cholesky factorization of a sparse matrix into a sequence of partial factorizations of dense smaller matrices (the so-called *frontal* matrices). To determine these frontal matrices the *elimination tree* must be pre-determined.

Here, following the hierarchy of nodes determined by the recursive discretization, the frontal matrices can be constructed without the need of pre-determining the elimination-tree, as the recursive discretization itself directly determines the elimination tree:

THEOREM 4.2: *"The triangles hierarchy built by the recursive triangulation algorithm of section 3 determines univocally the elimination tree corresponding to the nodes belonging to each initial triangle"*

Proof: it is sufficient to demonstrate that 1) nodes that are on the vertices or on the edges of a layer of triangles are *parents* (in the elimination tree) of the internal nodes, and that 2) nodes that are internal to different triangles of the same upper layer belongs to different subtrees. We refer to [85] for the construction of the *elimination tree* corresponding to a *permutation matrix*. Now let us prove point "1)". Following the nodes elimination order defined by a *permutation matrix*, *children* nodes come first than their *parents* and *parents* nodes receive the updates only from their *children*. Now we follow the backward elimination order that arises by construction of the recursive discretization here considered. Here, nodes are eliminated level-by-level starting from the last one, i.e. nodes that are on the vertices or on the edges of a layer of triangles cannot be *children* of internal nodes. Moreover, nodes that are on the vertices or on the edges of a layer of triangles receive updates from the internal nodes (i.e. they are *parents* of the internal nodes). Now let us prove point “2)”. By construction, nodes internal to a triangle of a certain layer do not update nodes internal to another triangle of the same layer. Hence they belong to different subtrees, whose roots are the interface nodes of that layer of triangles. ♢

Figure 28 shows a simple example: it is represented the recursive refinement inside an *initial triangle* and the corresponding elimination tree obtained as indicated in [85]. The numbers indicate the elimination order induced by the recursive discretization. The figure shows that it is the same that would be followed using the *elimination tree*. Moreover, since the construction of the *frontal matrices* can be done independently inside each *initial triangle* (see section 3.2), few frontal matrices must be kept in memory at the same time, hence it can be avoided what is recognized to be a major drawback of the multi-frontal method with general re-ordering, i.e. the unpredictable and sometimes excessive demand of computer memory for storing the stack of frontal matrices [84].

4.3.3 Application to Adaptivity and to Problems with Arbitrarily-Varying-Coefficients

The recursive meshing algorithm presented in Section 4.3.1 can be driven also by a-posteriori error estimates. This means that the hierarchy will be updated during a refinement which in this way may be done not only on the last level of triangles (as in the a-priori version of the algorithm) but also on the previous ones.

The flexibility given by the *nodes insertion step* of the algorithm permits the construction of general adaptive meshes. The un-refinement operation is simplified by the tree-like hierarchy of the mesh. A complete application to an adaptive solver is under development.

Problems with arbitrarily varying coefficients are, for example, problems with internal interfaces in the domain and/or with non-homogeneous coefficients. Let us consider first the case of non-homogeneous coefficients:

$$Lu(x,y) = -\frac{\partial}{\partial x}\left(k(x,y)\frac{\partial}{\partial x}u(x,y)\right) - \frac{\partial}{\partial y}\left(k(x,y)\frac{\partial}{\partial y}u(x,y)\right) = f(x,y) \tag{4.1}$$

$$u(x,y) = 0 \qquad \forall (x,y) \in \partial\Omega \qquad (x,y) \in \Omega \quad \text{and} \quad \Omega \text{ is a bounded domain} \subset R^2$$

where f is a function in $L^2(\Omega)$ and $k(x,y) \in L^\infty(\Omega)$ satisfies

$$0 < \alpha \leq k(x,y) \leq \beta < \infty \qquad \forall (x,y) \in \Omega$$

.

All functions and vector spaces considered in the following are real. The coefficient $k(x,y)$ can vary roughly or be highly oscillating on Ω. If it varies roughly, then the solution

u will also be rough (see [10] for further important considerations on finite elements approximation for this kind of problems). Various strategies are adopted in the finite elements literature for solving a problem of this kind. The first we mention is that of classical homogenization [18], where an homogenized PDE is obtained for this class of models (in one dimension for simplicity):

$$-\tfrac{d}{dx}\left(k(\tfrac{x}{\varepsilon})\tfrac{d}{dx}u^{\varepsilon}(x)\right) = f(x) \qquad x \in [0,1]$$

is substituted as $\varepsilon \to 0$ with $-\frac{d}{dx}\left(a_0\frac{d}{dx}u^0(x)\right) = f(x)$, that constitutes the homogenized problem. Since on the homogenized PDE the coefficient is constant, there is no more need to follow material interfaces or coefficient variations with the finite element discretization. This method works only if $\varepsilon \ll 1$ (that is the scale of variation for the coefficient and the problem's scale are well separated) and $k(x)$ is periodic or semi-periodic. A second approach, present in the literature, is to adopt special Finite Elements; in [10] are used trial functions spaces which have embedded the coefficient variations:

$$S_h = \left\{v \in L^2(\Omega) : v|_T \in \text{span}\{1, \textstyle\int_0^x \frac{dt}{k(t)}\text{dt}, y\}\right\}$$
$$\forall T \in C_h \text{ , } v \in C^0 \quad \text{on the nodes , } v = 0 \quad \text{on}\Gamma_\Omega$$

Here again it is not necessary to fit the interfaces with the finite elements, but the coefficient can vary sharply in at most one direction. A third approach is that of deriving an explicit formulation of the element with inhomogeneities inside; in [88] is developed the eight-noded rectangular element with a circular inhomogeneity at its centre. This approach seems to be a good solution if the variations of the coefficients can be characterized in terms of a specific class of inhomogeneities of the material.

Let us consider now problems in which sharp interfaces between different materials are present. If problems of this kind are solved using finite elements, it is well known the importance of placing nodes at sharp interfaces (and edges possibly overlapped to the interfaces) when these are present in the problem domain. In fact [97], let us consider the model problem

$$\nabla \cdot (p\nabla u) = f \qquad in \quad \Omega$$

$$u = 0 \qquad on \quad \partial\Omega$$

where

$$p = \begin{cases} p_1 \in \Omega_1 \\ p_2 \in \Omega_2 \end{cases} \qquad \Omega_1 \cup \Omega_2 = \Omega$$

The classical formulation of the problem requires that the PDE holds separately in Ω_1 and Ω_2, with u and $p\dfrac{\partial u}{\partial \nu}$ continuous across the interface (ν is the normal direction to the interface):

$$p_1\frac{\partial u}{\partial \nu}\ |_{\Gamma^-} = p_2\frac{\partial u}{\partial \nu}\ |_{\Gamma^+} \tag{4.2}$$

This expression comes out from the Green's formula (integration by parts, in one dimension); it is a *natural* boundary condition for the problem (i.e. it is not satisfied a-priori by the approximating functions). It can be said that the continuity of the first normal derivative across the interface is weakly enforced.

It turns out that the standard error estimates for finite elements, which rest on an assumed smoothness of u, will degenerate at the interface, unless a node is placed on the interface. This permit the approximating functions to copy the jump condition (4.2). In fact, as long as the approximating functions are not forced to have a continuous derivative, and thereby violate the jump condition, the approximation will be good.

For both of these problems (i.e. non-homogeneous coefficient and interfaces between different materials) the recursive meshing algorithm presented in this Chapter places automatically nodes to describe the parameter variations or at the interfaces, if present. More precisely, it places nodes in such a way that into each element the varying coefficient is almost constant (i.e. varies from a constant value within a prescribed narrow tolerance). This constant value can then vary arbitrarily from element to element, as dictated by the problem. We will consider for convenience linear triangular elements. Let $T_k = \{e_K\}$ be a triangulation of the domain Ω satisfying the standard assumptions [42] and also the condition that $k(x, y) = a_k \quad in \quad e_k$; then

$$\int_\Omega \frac{\partial}{\partial x}\left(k(x,y)\frac{\partial}{\partial x}u(x,y)\right)d\Omega = \sum_{e_k \in T_k} \int_{e_k} a_k \frac{\partial^2 u(x,y)}{\partial x^2} de_k = \sum_{e_k \in T_k} a_k \int_{e_k} \frac{\partial^2 u(x,y)}{\partial x^2} de_k$$

as the a_k is constant within each element. Hence the interfaces and variations of the coefficient are fitted automatically by the discretization algorithm and the algorithm itself can be applied without any restriction on coefficient's variation. This approach, in general, could suffer from the fact that the number of degrees of freedom required to solve the problem can be large, with subsequent high requirement of memory and computing power. For this reason the algebraic properties of this triangulation algorithm are useful.

Let us reconsider the application to a shallow water problem cited before and presented in Figg. 23 24 25. The shallow waters problem is solved using a splitting technique in which water elevation is governed by an equation of the type (4.1). See [34] [36]. With the application of this algorithm to that problem it is possible to guarantee that each triangle lies on a portion of the domain at constant depth. In this way the integrals necessary to the computation of the element matrix are greatly simplified.

PART III

Solution Update

The adaptivity in the Finite Element Method creates also a main problem of numerical linear algebra, i.e. the update of the solution of the linear system (or of the eigenvalue problem, a case not considered in this Thesis) after a mesh adaptation.

The adaptivity involves *local* operations on the finite elements discretization. The a-posteriori error estimates must be local error estimates, the data (e.g. the *residuals*) are local and the procedure is local (e.g. local problems solving, local averaging, ...), cfr. Chapter 2. The adaptation of the mesh is generally local and local data manipulation are generally required to perform a refinement/un-refinement step. In this Part of the Thesis it is shown that also the update of the solution can be seen and solved as a local numerical process, although the solution is obtained by a system of (coupled) linear equations.

In the literature, this locality of the solution-update problem has been received little attention and there are not available special algorithms for this problem. This is more true for *direct methods* than for *iterative methods*.

This part of the Thesis addresses this issue. If we consider that the machine-time spent to solve the linear system of equations is a big percentage of the total time required to get the finite element approximation, it turns out that this issue is quite important (as is demonstrated e.g. by the quantity of resources dedicated in the last years by the research community to the development of faster algorithms for the solution of large linear systems of equations, which primarily comes from the discretization of partial differential equations)

In general, it is important to say that the refinement/un-refinement process is usually a trial-and-error procedure, because a quantitative prediction of the effect of a refinement/un-refinement operation on the approximation error is lacking. Hence it is necessary to repeat the operation more times during an adaptive analysis. This fact augments the importance of an efficient linear algebra computational procedure, which becomes even more important if the problem at hand is time-dependent and refinements/un-refinements must be done every few time-steps.

From the linear algebra point-of-view, the degrees-of-freedom associated with a finite-element mesh correspond to a set of equations in the resulting linear system to be solved, and the modification of the mesh performed by a refinement/un-refinement operation corresponds to modify and add/remove equations to/from the system.

The standard approach used in Adaptive Finite-Elements Analysis is to re-compute, instead of updating, the solution at each refinement/un-refinement step [100], throwing away the computations made at the previous step.

Direct methods are effective on updating modified problems where the mesh does not change (so that the coefficient matrix remains the same) and where there are multiple right-hand-sides. For instance, in linear stress analysis the response to many different load cases may be of interest. It is weel-known that, having computed the factorization of the system matrix, the solutions corresponding to different right-hand-sides are obtained quite cheaply. However, by definition of adaptive refinement, a mesh-modification (and hence a matrix perturbation) is surely made. In this case the previous factorization cannot be substantially re-used. In the Finite Element Analysis arise similar problems to this adaptivity solution-update problem in which a set of slightly modified problems, in which some entries of the system matrix (i.e. the so-called stiffness-matrix) are modified, must be solved. In [83] it is shown a technique for calculating the update of the solution from the solution of the previous system. This technique however updates always all the variables of the system. *Iterative methods*, like the the Preconditioned Conjugated Gradient, reduce the number of iterations according to the rank of the interface matrix (the *domain decomposition* idea [64]), but they work anyway on the overall problem and not locally, hence also the interface remains the global one.

These techniques update always all the system variables and we will show that this is in general inefficient. For this purpose, let us give a precise characterization of the problem.

Given a linear system of equations coming from a Finite Elements discretization, we will see that the adaptvity-update problem is to find the solution of this system after a perturbation of the matrix and right-hand-side plus an addition/subtraction of an certain number of columns and rows. Let us consider for example a refinement operation. Here $Ax = b$ was the original system, that after refinement becomes:

$$\begin{bmatrix} A + \delta A & \delta F^T \\ \delta F & A_r \end{bmatrix} * \begin{bmatrix} x + \delta x \\ \delta x_r \end{bmatrix} = \begin{bmatrix} b + \delta b \\ \delta b_r \end{bmatrix}$$

It can be re-written in the form:

$$\begin{bmatrix} A_a & G^T & 0 \\ G & A_i + \delta A_i & \delta H^T \\ 0 & \delta H & \delta A_r \end{bmatrix} * \begin{bmatrix} x_a + \delta x_a \\ x_i + \delta x_i \\ \delta x_r \end{bmatrix} = \begin{bmatrix} b_a \\ b_i + \delta b_i \\ \delta b_r \end{bmatrix}$$

where $F = [0 \quad \delta H]$, 'r' are the equations corresponding to added nodes by refinement, 'i' corresponds to the nodes of original elements which were subdivided by refinement and 'a' corresponds to the remaining nodes. This means that we need to perform an update of the solution for a locally perturbed system of equations of the form $(A+\delta A)*(x+\delta x) = b+\delta b$. In the following we will address this problem. After simple manipulations on the locally perturbed system, we have:

$$(A + \delta A) * \delta x = \delta b - \delta A * x$$

This system has, for the problems here considered, the interesting property that his right-hand-side vector is different from zero only in a limited number of elements (i.e. a small percentage of the total). In fact it is easy to demonstrate that these elements are only the degrees-of-freedom associated to the finite-elements that are modified or added by the refinement/un-refinement operation (i.e. the x_i and x_r variables of the previous equations). At this point two other considerations becomes important. The first one is that usually the refinement/un-refinement step modifies a small percentage of the total number of finite-elements of the mesh. The second one is that in principle the solution-update will be different from zero only in a neighbourhood of the nodes corresponding to right-hand-side vector components different from zero (cfr. Section 5.2.1). Combining all these considerations together, it may be argued that the solution will be modified (above a finite tolerance) only in a relatively small set of system variables. Hence it is desirable to factorize the related equations and compute the solution-update only for this restricted set of variables.

These considerations apply to any type of adaptive refinement/unrefinement (h-, p- or hp-) based on any type of a-posteriori error estimates [100], and to "mesh-smoothing" techniques [16] too.

Hence the aim of the following chapter is to analyze some possible ways of solving the system only for this subset of equations, using direct or iterative methods. The methods presented can be seen as accelerators of direct and iterative methods, for the solution of the adaptivity-update problem.

Moreover, a correlated aspect of this problem, is that for many practical reasons (outside numerical analysis) in a Finite Element analysis it is given a maximum *level of accuracy* attainable, in the sense that solution values which are below this level are not to be taken into consideration. In this case, a greater solution accuracy than that allowed by modelling and data accuracy, is in principle a waste of machine-time. Hence, an approximate solver with selectable accuracy is useful for finite elements applications (if its computational cost varies proportionally to the level of accuracy desired).

Iterative methods approximate the solution of a linear system up to a level chosen by the user, with a computational cost that varies (i.e. increases) with the level of accuracy required. *Direct methods* approximate the solution of a linear system up to the maximum level of accuracy attainable with the algorithm adopted, with a fixed computational cost.

An extension of the method here presented for the adaptivity-update problem may produce also an approximate direct solver. This is currently under development.

Chapter 5

5
System Solvers for Adaptivity

Let us start considering *Direct methods*, which have a long tradition of use also in solving linear systems arising from finite-element discretizations. The *fill-in* is generally recognized as the main responsible for the amount of computations and execution-time required when a direct method is applied to the solution of a linear system with a sparse matrix. Since the amount of *fill-in* produced during the factorization is strongly dependent on the equation-elimination *ordering* adopted, the last fifthteen years have seen a big research effort on finding the best ordering attainable at a reasonable cost. For one of the latest ordering algorithms available, see e.g. [48] on the Internet. The ordering problem is NP-complete, so current techniques use heuristics based on graph theory (see e.g. [52] and Section 4.3.2). Within the class of sparse matrices deriving from finite element discretizations, which is the scope of this paper, these techniques generally equal or outperform previous techniques. In the following we will see two acceleration approaches for the solution of the adaptivity-update problem and their application to a *direct method* and the possibilities to apply them also to *iterative methods*.

5.1. The Waves-Bordering Approach

Here we present the Waves-Bordering algorithm for the solution of linear systems of equations arising from the application of the Finite Elements Method. This method offers a formulation which can be used to solve the adaptivity-update problem with the efficiency indicated in the Part III introduction.

The approach takes its origins from the Bordering/Reverse-Bordering method for the recursive solution of linear systems of equations, in which equations can be added/removed and the corresponding solution recursively updated. The algorithm can be seen as a substantial modification of the Bordering algorithm [24]. This modification relies upon the fact that the system matrix is obtained from an operator dicretization on a finite elements mesh (actually the same reasoning could work also on a finite-difference mesh). Assuming that the Bordering algorithm is known, two are the main points for understanding the Waves-Bordering algorithm: bordering blocks selection criteria and modified solution mechanism.

First, the algorithm starts solving the equations (one or more) corresponding to a specified initial set of mesh nodes (i.e. components of the solution) assuming that all other components are zero. If these mesh-nodes plus their adjacent edges in the mesh constitutes

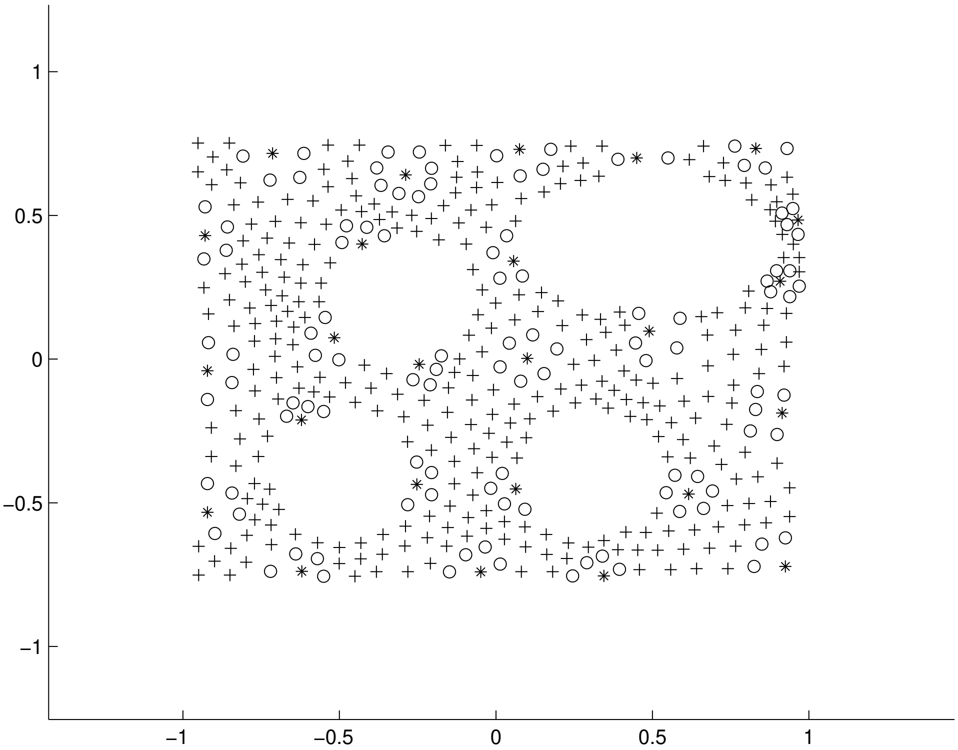

Figure 29: waves-sources (*), one-front independent waves (o) and the interface problem (+) generated for the test-case of Section 5.1.5 . The underlying mesh is hidden for better comprehension

a connected subgraph we speak of *Single Wave*, while if the resulting subgraph is not connected we speak of *Multiple Waves*, i.e. there is a finite set of independent connected subgraphs.

Then, successive groups of equations (i.e. *fronts*) are solved. Here the term *front* arises considering successive sets of nodes not previously considered and adjacent to the previous front (the initial set of nodes can be considered the first front, i.e. wave-source). The fronts here defined behave on a mesh like wave-fronts of propagating water-waves when an object has hit the surface in the initial set of nodes (see e.g. fig. 3.1 that represents the setting used for obtaining the results of Section 5.1.5 . Here the fronts are built from-inside-to-outside but it is possible to do the same starting from a contour and building fronts from-outside-to-inside, cfr. Section 5.1.3 . In the Waves-Bordering algorithm there is a one-to-one correspondence between fronts and bordered blocks of equations.

Secondly, the algorithm computes via block-bordering the solution for the newly bordered variables but does not update the previous ones (this constitutes the “forward-run”). When the last block of equations is added, for the corresponding variables the computed values are the exact solution in the overall system. Hence they are now known and can be substituted back, modifying the right hand sides of the previous equations. Due to the way of choosing blocks, previously outlined, only the right hand sides of the equations corresponding to the previous added block have to be modified. Hence re-computing via block-bordering the previous block with modified right hand side, it holds the true solution for the variables corresponding to that block. Iterating this process to the first block holds the complete solution (and constitutes the "backward run"). As we will see in the next subsection and in the experimental results, this modification lowers considerably the computational cost of the method.

5.1.1 Single Wave

Let us start considering a finite element discretization of a 1-D problem. Here there is a single path of propagation, each front consists of only one node and, with a convenient node-numbering assignment, the resulting matrix is tri-diagonal. We do not loose in generality if we suppose that the wave origin is located at the first boundary node, i.e. corresponds to the first variable of the solution vector. Referring to [24] for the notation, the Wave-Bordering algorithm works as follows:

- forward run:

  $z_{k+1}(k+1) = \frac{f_k - v_k * z_k}{\beta_k} \qquad k = 1 \dots N-1$

  where $\beta_k = a_k - v_k A_k^{-1} u_k$

- backward run:

  $z(k) = z_k(k) - \frac{v_{k+1} * z_{k+1}}{\beta_k} \qquad k = N-1 \dots 1$

Two things it is important to note: the first is a substantial simplification of the computation of the β_k with respect to the Bordering algorithm: since v_k and u_k have only the last component different from zero, we have $\beta_k = a_{(k+1,k+1)} - a_{(k+1,k+1)} * a_{(k+1,k+1)}/\beta_{k-1}$ where the a's are now A_{k+1} matrix entries. The second is that the β_k of the backward run are exactly the same as those of the forward run and must not be re-computed. This algorithm solves the system with a $O(N)$ number of floating-point operations.

A natural extension for the 2-D and 3-D settings of this algorithm is to consider the Block-Bordering equations [24] and identify bordering blocks with the *fronts* previously defined. In this way are retained both the possibility of doing the simple two-run method and the simplification on the calculation of β_k. The only substantial difference is that now β_k is a matrix, not a scalar, and must be inverted for computing β_{k+1}. Moreover, thought a_k is almost a tri-diagonal matrix, $v_k \beta_k^{-1} u_k$ is full and hence β_k is also full. Now, we have $\beta_k = a_k - v_k A_k^{-1} u_k = a_k - [v_k] \beta_k^{-1} [u_k]$, where $[v_k]$ are the last p_{k-1} columns of v_k and $[u_k]$ are the last p_{k-1} rows of u_k. The algorithm becomes as follows:

- forward run:

  for $k = 1 \dots N-1$ solve the following sub-problems (unknowns are in bold): $L_{k-1}\mathbf{Y} = P_{k-1}[u_k], \qquad U_{k-1}\mathbf{X} = Y, \qquad \mathbf{S} = [v_k]X, \qquad \beta_\mathbf{k} = a_k - S, \qquad \mathbf{P_k}\beta_k = \mathbf{L_k U_k}, \qquad L_k\mathbf{y} = P_k f_k$, and finally $U_k \mathbf{z_{k+1}}(\mathbf{i_{k+1}}) = y$

  where i_k are the indices of the i-th block of variables.

- backward run:

  for $k = N-1 \dots 1$ solve the following sub-problems (unknowns are in bold): $L_k\mathbf{y} = [v_{k+1}]z_{k+1}(i_{k+1}), \qquad U_k\mathbf{x} = y, \qquad \mathbf{z(i_k)} = z_k(i_k) - x$

Note that explicit β_k matrix inversion is avoided as suggested in [64].

Hence in 2-D and 3-D problems the front is a set of simultaneous equations whose matrix is ricursively derived from the inverse of the matrix corresponding to the previous front. For this fact it is desirable that fronts does not grow too much, i.e. the number of variables that belong to a front be limited as much as possible.

5.1.2 Multiple Waves

The opportunity of parallelizing the method and the requirement of small fronts lead to the concept and implementation of *multiple waves* and therefore multiple fronts propagation.

The technique adopted to takle with multiple waves is very important because it is a good way to keep limited the largest length of the resulting fronts and, as already mentioned, the method described above is cost-effective only if the fronts are not too crowded, in the sense of too many equations per block and too big matrix to be inverted. Multiple waves have two interesting properties that must be accounted for.

The first one is that they permit to solve N equations working in parallel on n waves whose fronts correspond to sets of uncoupled equations and therefore they can be solved independently of each other (with also the advantage of taking the inverse of n matrices of dimension $(\frac{N}{n}) \times (\frac{N}{n})$ instead of one matrix of dimension $N \times N$). In this way the length of the fronts can be limited a priori by the geometry of the finite elements and by the distance between waves sources, computed in terms of mesh edges. Moreover, the consideration of multiple waves in this way brings us to a method that is intrinsically parallel.

The second property is that when two (or more) fronts belonging to different waves encounter, they join themselves in a unique front described by easily summing (with a proper shift to the indices) their corresponding matrices. Let us see a simple example of two joining fronts:

let x_1, x_2 be (the variables corresponding to) the last fronts of two disjoint waves and x_3 be (the variables corresponding to) the joining front; then x_1 and x_2 are the last two blocks of equations bordered to the overall system prior to the joining front. Now if we join x_2 to x_1 through x_3 we obtain β_k matrices equal to $\beta 1_k = B1_k$ and $\beta 2_k = B2_k - [v_k]\beta 1_k^{-1}[u_k] = B2_k$, because x_1 and x_2 are disjoint and hence $v_k = u_k = 0$. Here Bi_k is the front-update matrix (i.e. β_k) for x_i and derived from solving previous fronts in the i-th wave, $i = 1, 2$. Now, when bordering the equations corresponding to the x_3 variables, we have:

$$\begin{bmatrix} B1_k & 0 & u31_k \\ 0 & B2_k & u32_k \\ v31_k & v32_k & a3_k \end{bmatrix}$$

and

$$\beta3_k = a3_k - \begin{bmatrix} v31_k & v32_k \end{bmatrix} \begin{bmatrix} B1_k & 0 \\ 0 & B2_k \end{bmatrix}^{-1} \begin{bmatrix} u31_k \\ u32_k \end{bmatrix}$$

$$= a3_k - \begin{bmatrix} v31_k & v32_k \end{bmatrix} \begin{bmatrix} B1_k^{-1} & 0 \\ 0 & B2_k^{-1} \end{bmatrix} \begin{bmatrix} u31_k \\ u32_k \end{bmatrix}$$

$$= a3_k - (v31_k B1_k^{-1} u31_k + v32_k B2_k^{-1} u32_k)$$

Note that this simple result holds because the inverse of a block-diagonal matrix is still block-diagonal and it can be easily extended for a front that joins n disjoint waves, with $n > 2$.

In practice it is convenient that the system be solved by a high number of independent waves joined by a single interface-block of equations. The interface-block tipically turns out to be far the longest front to be computed. It can be computed with a direct or iterative method or applying to it again the Waves-Bordering algorithm. The matrices that appear in the front-update systems of equations for the independent-waves fronts are dense and the matrix corresponding to the interface-block is sparse with a block-diagonal structure (with several small blocks) plus few entries outside. We could say that the Waves-Bordering method with multiple waves condense with few operations the original sparse system into $n >> 1$ small dense systems plus a sparse system (the interface problem) of dimension much smaller than the original one.

For these reasons, when considering the refinement/un-refinement problem we do not use perturbed nodes as wave origins (as could be suggested from what said in Sections 2 and 4), to let the method working in his optimal operational conditions. Note that this is increasingly important as the refinement of the mesh increases, because then the propagation of the solution-update interests a growing number of nodes, since the distance between nodes (i.e. the length of the elements edges) is diminished, and for this reason the fronts become increasingly crowded of variables. Instead, number and locations of waves-sources, number of fronts and the interface-block itself are all decided at the beginning. Note that the number and positions of wave-sources should be preferably updated (for the efficiency of the algorithm) after a refinement/unrefinement operation, but it is not mandatory.

5.1.3 Domain Decomposition

The *multiple waves* approach seen in the last Section requires to solve a problem of domain decomposition. Here with this term we refer to the technique widely used of decomposing the algebraic problem into smaller subproblems which can be solved quite independently [94]. Since we refer to *direct methods*, this involves a successive *block-elimination*, while for *iterative methods* this involves a *block iteration*. The domain decomposition formulation here becomes particularly difficult because the solution of the next *front* requires a Schur's Complement computation and this becomes too demanding if the length of the front grows too much. Moreover, as will become clear in the following, the solution of the adaptivity-update problem becomes efficient only if the domain has a sufficiently fine subdivision.

An immediate realization of this requisite brings to the formulation of a large interface problem. This is in general a disadvantage, since with the existing domain decomposition techniques (in the sense indicated above) the matrix associated to this problem is usually much denser than the original matrix. This fact, if the interface problem contains a considerable percentage of the total equations, creates also a considerable amount of *fill-in* during the factorization.

Again, since the computational cost should be preferably lower than it is generated by the best existing general-purpose direct solvers, i.e. the amount of *fill-in* created must be not bigger than that created by the best *pre-ordering algorithms*, it must be taken into account the possibility of including these pre-ordering algorithms in the formulation of the domain decomposition or at least in the solution process.

By consideration of these constraints and opportunities, they have been tried three domain-decomposition formulations, which are presented in the sequel.

5.1.3.1 Graph-Partitioning

One reasonable method to block-partition a sparse matrix arising from a finite elements discretization is to partition the graph associated to the finite element mesh. This is because to each node of the graph it is associated one equation in the system of equations to be solved. Well established algorithms exist which solve a graph-partitioning problem (see e.g. the theory and the algorithms contained in the METIS library [89], which has been used in this Thesis).

With this approach it is straightforward to obtain a balanced block-partitioning of the matrix into a desired number of blocks.

The experiments indicate that there is a general problem with this approach, and precisely that the existing techniques of graph-partitioning give a balanced partition (i.e. all subdomains are almost of the same size) but they do not take into account the *degree* (i.e. the number of neighbours) of the nodes to do the partition, which is instead the base for the success of the *minimum degree* family of algorithms. It turns out that better results can be achieved compared to the *Minimum Degree Algorithm* [63], available in MATLAB, but not better at all if compared with the newer *Approximate Minimum Degree* algorithm [3], available at [48].

5.1.3.2 Pre-ordering

The pre-ordering algorithms for sparse matrices (see [52] for an overview) produce an elimination ordering that tries to minimize the *fill-in* produced during the factorization of the matrix. The best performing algorithms seems to be those deriving from the *minimum degree algorithm* and the most recent implementations are here considered.

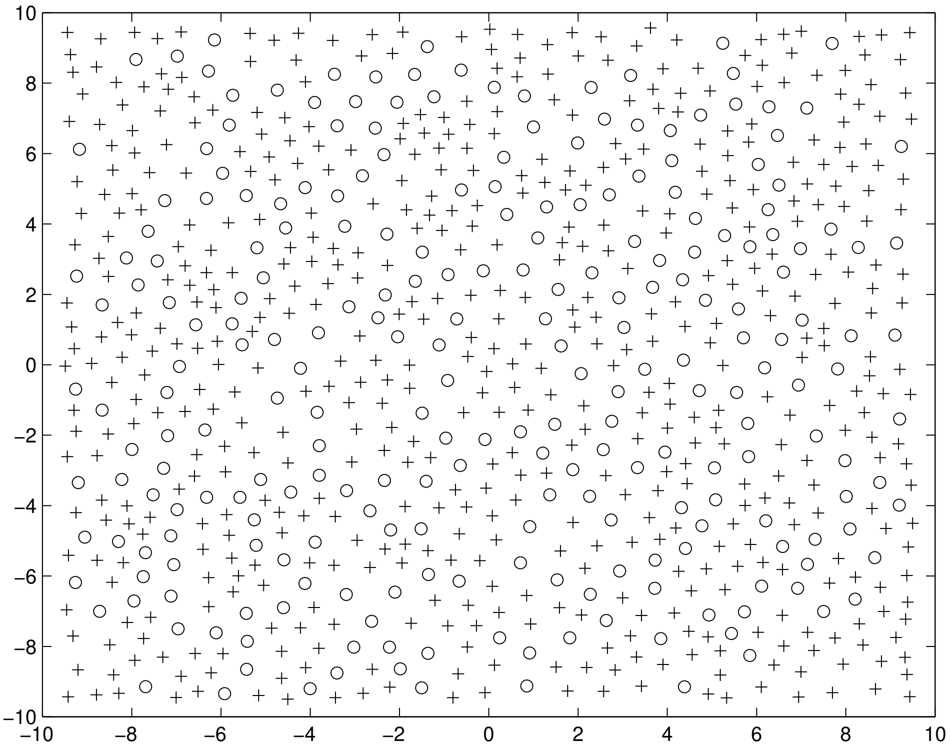

Figure 30: domain decomposition created by the first levels of the elimination tree determined by the Approximate minimum Degree algorithm (with a 'o' are indicated the nodes which constitutes the interface problem)

The elimination ordering produced by these algorithms can be represented also by a tree-hierarchy (the *elimination tree*), that gives the possibility of doing the elimination in parallel, since it holds that distinct sub-trees can be eliminated independently from each other. This is actually done in the well-known *multi-frontal* codes [85].

From the elimination-tree it can be devised a domain decomposition by considering an appropriate number of levels in the tree (starting from its root) as the interface problem and the remaining levels as the subdomain. Figures 30 and 31 show this situation for two different choices for the subdivision of the tree-levels. In this way, the fill-in will be that produced by the ordering algorithm adopted (i.e. the best available), but it can be noticed that the shape of the subdomains is highly irregular and the number and average-dimension of them cannot be decided a-priori.

A second advantage of this approach, however, is that here the solver for adaptivity can be easily embedded in an existing multi-frontal code.

5.1.3.3 Recursive Discretization

The recursive meshing technique presented in Chapter 4 can produce a successful domain decomposition for direct methods which is suited also for the adaptivity-update problem.

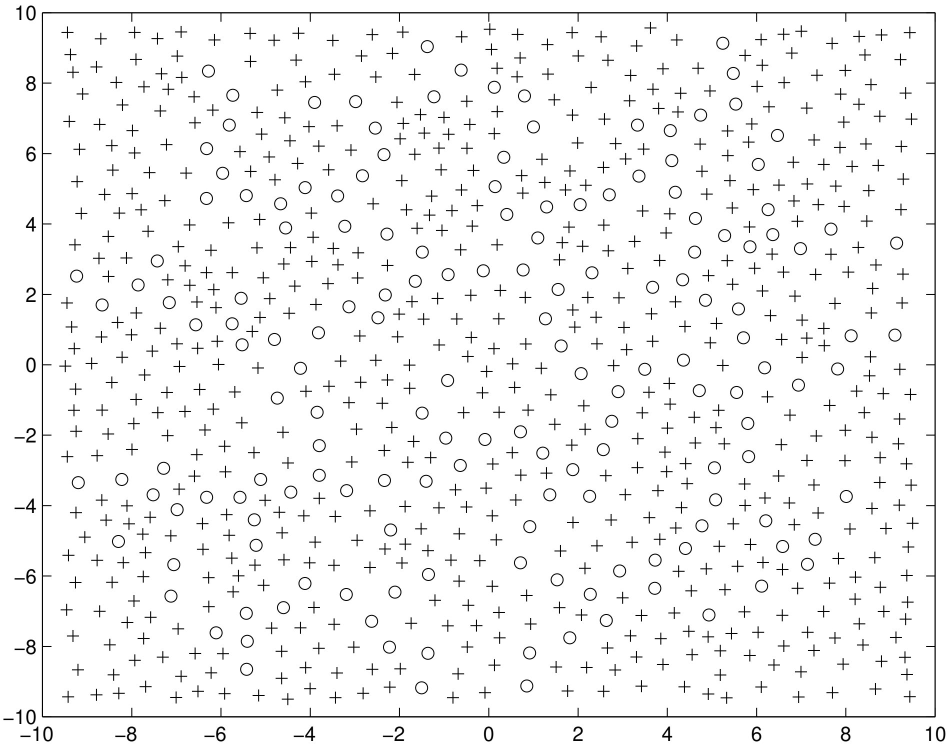

Figure 31: domain decomposition created by the first levels of the elimination tree determined by the Approximate minimum Degree algorithm (with a 'o' are indicated the nodes which constitutes the interface problem)

In this case, the *initial* plus *level-1* nodes constitute the interface problem, while the others constitute the subdomains. With this approach the number of subdomains can be easily controlled when defining the *initial* mesh.

In Chapter 4 we have seen that the natural elimination ordering determined by this recursive discretization produces a *fill-in* comparable with the best ordering algorithm for the problem of eliminating the *internal* nodes. Here we see in the following Tables that this is even better for the overall matrix factorization problem.

In the following Tables, are presented the results of the full factorization of the matrix arising from the application of the recursive algorithm to the same problem which did produce the Tables presented in Chapter 4. With "FEMRA" we mean the natural ordering obtained from the recursive discretization algorithm. The values of the parameters and the recursion levels are the same. For the first Table was chosen $\Delta C = 5.0$:

method	2 levels		5 levels		7 levels	
FEMRA	12784	164%	13724	140%	13898	132%
COLAMD	14358	184%	17101	174%	18032	172%
SYMMMD	15750	202%	19697	200%	21030	200%

and for the second Table it was chosen $\Delta C = 7.0$:

method	3 levels		5 levels	
FEMRA	11051	166%	11207	159%
COLAMD	12196	183%	12636	179%
SYMMMD	13274	199%	14412	204%

5.1.4 The Approximate Waves-Bordering Algorithm

The Waves-Bordering method, previously presented, solves all the equations of the system. To solve the adaptivity-update problem it can be approximated as follows [37]. Having in mind the Finite Elements formulation of the problem at hand, the perturbation to the right-hand-side vector components corresponding to the refinement/un-refinement modified set of mesh nodes acts like a localized forcing term in the discretized weak form of the differential operator. Hence, thinking that the solution update propagates starting from these nodes is physically and mathematically correct. Taking into account the way with which the Waves-Bordering method calculates the solution (i.e. fronts propagation), it is desirable to match the solution-update propagation and the update-computation propagation and stop the algorithm when the solution-update vanishes. This is roughly what is done by the Approximate-Waves-Bordering algorithm. Here the term “Approximate” arises because the algorithm approximates the solution-update up to a user-defined threshold.

5.1.4.1 Neglecting the Back-Propagation

Let us consider a discretization of a one-dimensional problem using finite elements; as is well-known, with consecutive node-numering it results in a tri-diagonal matrix. See it in an example:

$$\begin{bmatrix} 1 & a_{1,2} & & & & 0 \\ a_{2,1} & 1 & & & & \\ & & \ddots & & & \\ & & & 1 & a_{n-2,n-1} & \\ & & & a_{n-1,n-2} & 1 & a_{n-1,n} \\ 0 & & & & a_{n,n-1} & 1 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Now, if the solution holds $x_i = 0$, than will be also $x_j = 0 \quad j > i$. This can be seen examining the equations that consitutes the bordering method [24]: given x_i the solution of the subsystem $[A_m, n] * [x_n] = [b_m] \quad m, n \leq i$ and adding the $i+1$ row and column it results the following value for the $i+1$ variable:

$$x_{i+1} = \frac{0 - a_{i+1,i} * x_i}{1 - a_{i+1,1..i} * A^{-1}_{1..i,1..i} * a_{1..i,i+1}} = 0 \tag{5.1}$$

The same result obviously holds for all $x_j \quad j > i$. It must be noted that the Bordering method performs updates of the solution taking into account only the contributions coming from the variables of the previous fronts (in fact, at each step i it considers all variables of the i-th bordered system). Then it is not important to consider the contributions coming from remaining variables if contributions from the previous fronts bring to a zero update, because all successive updates will be zero. The same happens in 2-D and 3-D.

Hence, if we want to determine the subset of variables for which the update is greater than zero, the hypothesis of neglecting the back-propagation is demonstrated to be correct. But what happens if we want to determine the subset of variables whose update is in modulus greater than a positive threshold ? Using again the Bordering algorithm equations some useful majorizations can be derived, but now they depend on all matrix coefficients (i.e. also from forthcoming fronts) and thus some smoothness hypothesis must be imposed to the coefficients. Moreover in 2-D and 3-D the situation complicates considerably. For these reasons it is better to specify the stopping criterion in terms of the residuals, like usually done by iterative solvers.

5.1.4.2 Solution-Update Approximation

The fundamental difficulty in obtaining in an efficient way the approximation of the solution-update is given by the fact that the mesh-zone (i.e. the set of nodes) to be updated is a-priori unknown, except for simple cases. The Waves-Bordering method let us determine it during the solution process.

In fact, starting from the nodes of the refined mesh region and applying the considerations on the solution propagation made at the beginning of Section 5.1.4, with the Waves-Bordering the updating zone can be enlarged re-using in part the previous computations done on the smaller zone. This can be easily seen from the equations of Section 5.1.1: adding more equations imply a moderate overhead for determining the front-update matrix in the forward-step and a small added-cost for updating the previous variables in the backward-step.

Now, the actual implementation of the AWB must also take into account what has been said in Sections 5.1.2 and 5.1.3 and relative subsections, mainly about the multiple waves, the interface problem, the incapsulation of the previous fronts and the amount of *fill-in* produced.

Hence the AWB here proposed starts from the WB setting (determination of number and position of wave-sources, fronts of the independent waves and interface-block) and applies these concepts also to the interface problem itself, starting from the refined region, i.e. it subdivides the interface-block in overlapping sets of nodes, which incapsulate also the subdomains.

In substance, the AWB solves also the interface problem adopting the Waves-Bordering concepts (in particular the wave-sources are now coincident with the refined/un-refined regions of the mesh) because this allows us to enlarge the updating zone and re-using in part the computations done for solving the previous part of the interface-block. The pieces of interface-block to be added correspond to some of the sets determined above.

When the residual has gone under the user-defined threshold, the algorithm stops. For the residual to be calculated, the solution must be updated through the backward-step, but this is a little added cost, compared to the cost of the forward-step. To avoid this over-head, the forward-step solution figures can be monitored; heuristic rules based on this criterion give some practical advantage.

5.1.5 Numerical Experiments

In this Section we show few numerical results that are indicative of the AWB properties when it is used in a refinement/un-refinement update problem. Let us start with a comparison between the Waves-Bordering algorithm (WB) and the Bordering/Block-Bordering algorithm. Tab 5.1 shows the number of floating-point computations per equation (i.e. per mesh-node) required for solving the system of equations arising from the finite elements discretization of a laplacian differential operator on the mesh that is represented in figure 29. The domain is rectangular with four holes inside. The number of equations is 439.

Waves-Bordering	Bordering	Block-Bordering
316	71265	100840

Tab 5.1

The next table, Tab 5.2, shows the behaviour of the Approximate Waves-Bordering algorithm when there is applied an increasing value for the perturbation, that affects an increasing number of nodes above the user-defined threshold. The comparison is made between the AWB algorithm and a Jacobi Preconditioned Conjugated-Gradient algorithm. Here the stopping threshold for the residual is the same for the AWB and CG algorithms (i.e. 10^{-2}). Figures are again in floating-point computations per equation.

nodes to be updated	AWB	CG
58	27	309
84	42	364
106	59	420
133	84	505
167	90	562
200	170	677

Tab 5.2

For the same problems, the costs (expressed in floating-point computations per equation) of Waves-Bordering (WB), Jacobi pre-conditioned Conjugated Gradient and Minimum-Degree pre-ordered Cholesky algorithms were, respectively, 316 , 2409 and 518 to update all the variables with the same resulting residual, 10^{-15} .

From the Table it can be seen that AWB reduces more the number of computations requested than CG does, as the number of nodes to be updated becomes lower (situation quite common in successive refinements/un-refinements).

5.2 The GreenLU Approach

The aim of this Section is to introduce, in a direct solver of systems arising from finite elements discretizations of linear initial/boundary value problems, the information about the underlying differential problem concerning the solution propagation.

This is not present in existing direct solvers, since only the sparsity pattern is used by the well-known graph-based techniques that keep the fill-in as low as possible, and matrix entries are used only to keep stable the algorithm as much as possible (i.e. rows and columns permutations, see e.g. SuperLU [49] [50]). These graph-based techniques are of general applicability within sparse matrices factorization, and do not consider the kind of problem that has generated the matrix to be factored. In the literature it is put on evidence something close to this issue, i.e. the need of being able to "exploit the sparsity structure", which is evidently dependent also on the nature of the problem, but little results have been achieved on this direction.

The importance of using the information about the underlying differential problem stays, for the problem here considered, in the possibility of determining a-priori the set of variables which will have a significant update after the mesh adaptation. The knowledge a-priori of this set is important because it becomes straightforward to use the best preordering algorithm available, within this set of equations, without the over-head imposed by the enlargement procedure inherent in the Waves-Bordering method.

The method here presented takes into consideration the Green's functions associated to the problem, i.e. the solution obtained with an impulsive forcing term and homogeneous boundary conditions applied.

Let us consider the solution that it is obtained applying a *point load*, i.e. an impulsive right-hand-side, when homogeneous boundary conditions are imposed. In general such solution, that corresponds to the *Green's function* [95] associated to the problem, evolves propagating in space, starting from the point-of-application of the impulse. Typically, this solution will be negligible in regions that are sufficiently far from this point. The location and dimension of these regions depend on the type of differential operator considered, on the shape of the domain, on the point-of-application of the impulse, on the intensity of the impulse applied and on the approximation error tolerance accepted for the particular finite element analysis. It is important to note that the computation of all variables corresponding to degrees-of-freedom (e.g. mesh-nodes) contained in these regions, could be avoided.

The application of these considerations to a more general problem than this (*point load*), and precisely that arising from the update of the solution after a mesh refinement/un-refinement step still holds, since this problem corresponds to the solution of a linear system in which the right-hand-side is non-zero only for a limited number of entries, corresponding to a spatially closed subset of degrees-of-freedom. This is the argument of Section 5.2.1 .

5.2.1 Green's Functions and Superposition Principle

Trivially, the solution of a linear system with a null right-hand-side is zero everywhere. Let us consider the solution to the simplest perturbation to the right-hand side from the null one, i.e. when only one entry is non-zero, and call it the *system impulse-response*. Since the system of equations is linear, by definition it satisfies the *superposition principle*, and therefore we have that the solution of the system of N equations is the sum of the N solutions obtained applying one right-hand-side entry at a time.

If the linear system arise from the Finite Element discretization of a linear operator A, it is straightforward to verify that the system impulse-response coincides with the *Green's function* associated to the problem, i.e. the solution of a boundary value problem of this kind:

$$
\begin{aligned}
Au &= \delta \\
u &= 0 \quad on \quad \Gamma_D \\
\mathbf{n} \cdot \Delta u &= 0 \quad on \quad \Gamma_N
\end{aligned}
\tag{5.2}
$$

where the Dirac impulse is applied on a mesh-node and the boundary conditions are homogeneous.

These considerations are important because they show that the system impulse-response can be characterized, e.g. with the knowledge of the Green's function, independently from the finite element discretization adopted.

Moreover, this characterization holds independently also from the boundary conditions values, since it is easy to show that the solution of a problem with impulsive right-hand-side and general Dirichlet and Neumann boundary conditions can be expressed as a linear combination of the system impulse-responses (cfr. 1.14).

In conclusion, the superposition principle and the mechanism (which belongs to the finite element method) of including the (Dirichlet and Neumann) boundary conditions into the system of equations, allow us to consider the solution of the system arising from the discretization of a general linear initial/boundary value problem as the linear combination of *system impulse-responses*.

Hence, characterizing the *system impulse-response* (and this is quite simple, if an analytic expression of the Green's function is known) gives the possibility to have a majorization of the solution and therefore to determine the subset of equations that will have a solution above a pre-determined threshold. This subset must be considered for the solution of the system.

The Green's functions of many differential operators of interest in engineering problems have nice properties, e.g.:

- the position of their maximum coincides with the point of application of the impulse;

- their values are negligible in regions that are sufficiently far from the point of application of the impulse;

These properties suggest us, for a direct solver, to start eliminating the equations from this point and to do it taking into consideration how the solution propagates from the point of application of the impulse. These considerations are the mathematical support to the *propagation* approach presented at the beginning of this Part III.

In general, the characterization of the system impulse-response depends on the specific operator, on the position of application of the impulse and on the values of the operator coefficients.

Let us consider the following example, the boundary value problem: find u such that

$$-k \cdot \Delta u = f \tag{5.3}$$

with arbitrary Dirichlet boundary conditions applied. The Green's function for this problem on an unbounded domain is simply expressed by [66]:

$$G(\mathbf{x}, \mathbf{x_0}) = -\frac{1}{2k\pi} ln|\mathbf{x} - \mathbf{x_0}| \tag{5.4}$$

where x_0 is the point-of-application of the impulsive forcing term. Now, an important result (see e.g. [66]) states that the Green's function on a finite domain has the same singularity at source location $x = x_0$ as does the Green's function on an infinite domain. More precisely it holds that, in a finite domain:

$$G(\mathbf{x}, \mathbf{x_0}) = -\frac{1}{2k\pi} ln|\mathbf{x} - \mathbf{x_0}| + v(\mathbf{x}, \mathbf{x_0}) \tag{5.5}$$

where $v(\mathbf{x}, \mathbf{x_0})$ represents the effect of the boundary and it is the solution of the homogeneous problem

$$-k \cdot \Delta v = 0 \tag{5.6}$$

subject to the same boundary conditions applied to the problem 5.3. From the *maximum principle* for Laplace's equation it follows that in steady state the maximum solution value for problem 5.6 occur on the boundary, hence a majorization for $G(\mathbf{x}, \mathbf{x_0})$ can be easily found, since $v(\mathbf{x}, \mathbf{x_0}) = \frac{1}{2k\pi} ln|\mathbf{x} - \mathbf{x_0}|$ on the boundary. Numerical results for this example are presented in Section 5.2.2 .

5.2.2 The Partial Factorization

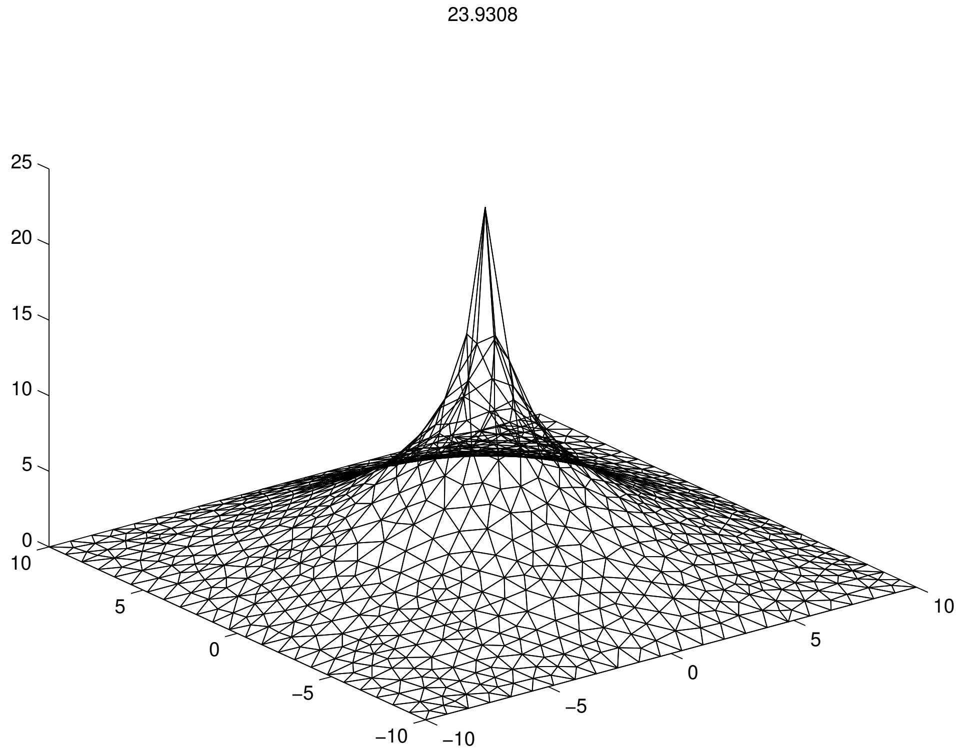

Figure 32: solution to a unitary impulsive forcing-term located in the center of a rectangular domain

In substance, the GreenLU method applied to the solution of an adaptivity-update problem makes a partial factorization of the corresponding system matrix, in the sense that only the subset of equations which corresponds to the variables that will have a value greater than a pre-determined threshold in the final solution, is eliminated.

This subset is determined by the application of the *Superposition Principle* and of the characterization of the *system impulse response* (cfr. Section 5.2.1). This characterization may be an analytic formula of the Green's function or, if this is lacking, a tabulated set of values.

Moreover, it is necessary to know easily the distances between each couple of nodes in the mesh. This can be obtained e.g. from the nodal coordinates, which are available by default on every finite element code.

Here follow the results obtained for solving the problem 5.3 on a rectangular domain and a finite-element mesh of 791 triangles. Three load-cases are considered, which consist of an unitary impulse applied in different points of the domain (Figg. 33, 34 and 32) . The results are contained in the following Tables, which shows in particular two quantities: the efficiency on finding the minimum set of equations to be updated ("updated/to-be-updated" in column 5), and the speed-rate obtained with the factorization of this subset of equations instead of the factorization of all the equations of the system ("LU-flops/GreenLU-flops"

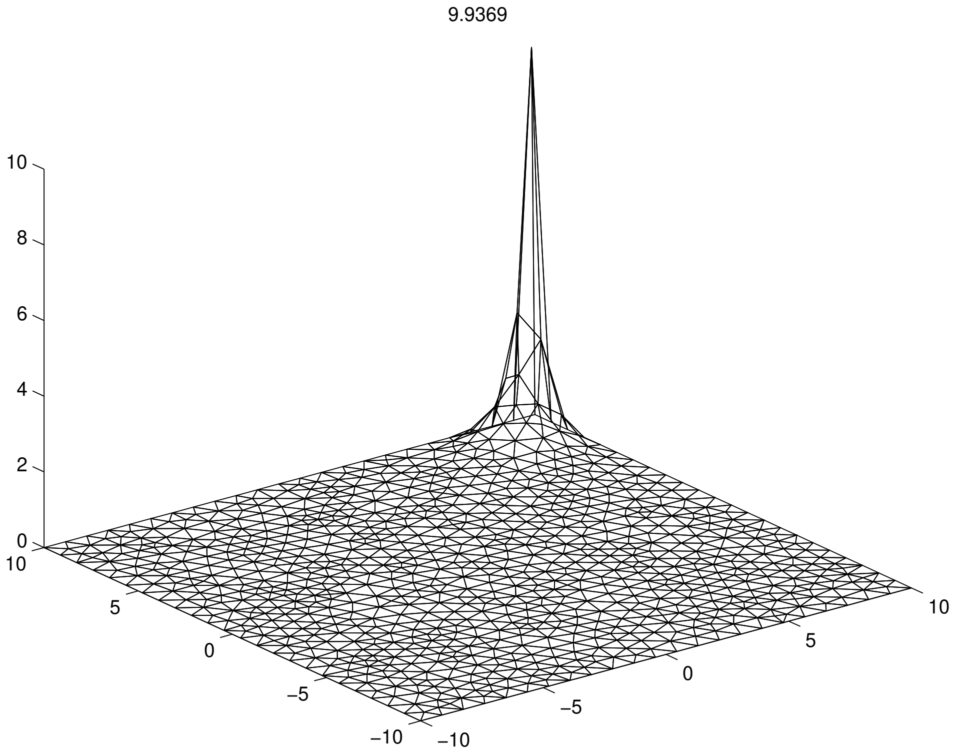

Figure 33: solution to a unitary impulsive forcing-term located near an angle of a rectangular domain

in column 7), which required 1492144 floating-point operations (flops).

The following Table corresponds to the application of the unitary impulse in the center of the domain (Figure 32) and a threshold $S = 3.5$ has been chosen:

k	to be updated	% tot	updated	$\frac{\text{updated}}{\text{to be updated}}$	GreenLU flops	$\frac{\text{LU flops}}{\text{GreenLU flops}}$
0.03	202	25.5	249	1.2	195429	7.6
0.05	85	10.7	99	1.2	44718	33.4
0.06	57	7.2	67	1.2	21886	68.2
0.07	32	4.0	40	1.2	8747	170.6
0.08	20	2.5	24	1.2	3644	409.5

The next Table corresponds to the application of the unitary impulse close to an angle of the domain (Figure 33) and a threshold $S = 0.2$ has been chosen:

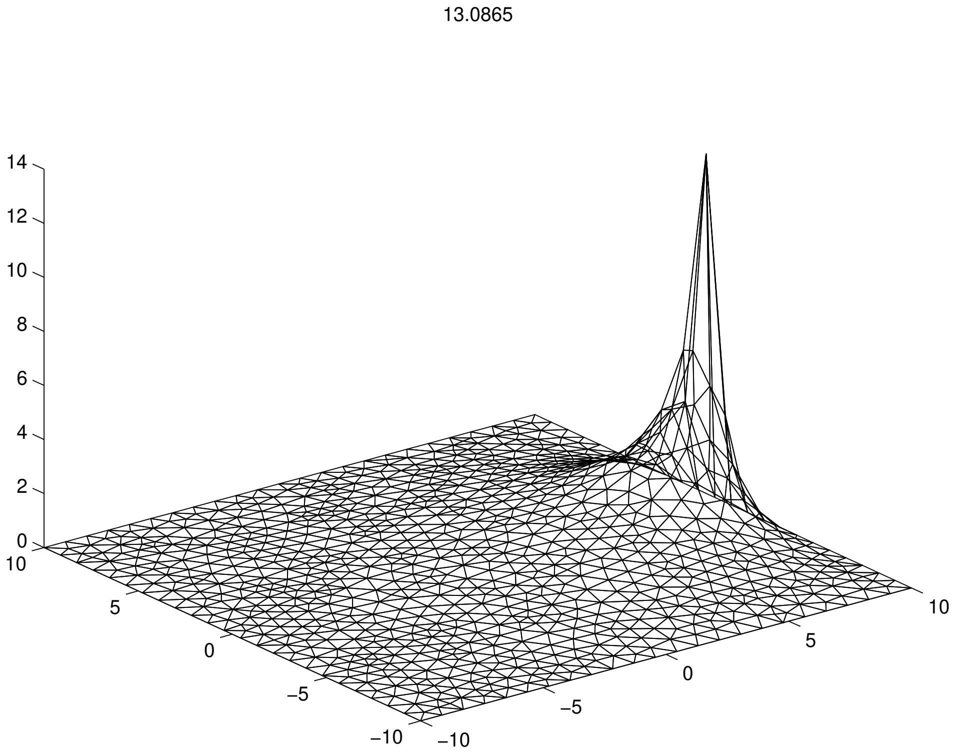

Figure 34: solution to a unitary impulsive forcing-term located near a mid-side of a rectangular domain

k	to be updated	% tot	updated	$\frac{\text{updated}}{\text{to be updated}}$	GreenLU flops	$\frac{\text{LU flops}}{\text{GreenLU flops}}$
0.03	33	4.2	40	1.2	7049	211.7
0.05	17	2.1	25	1.5	2955	505.0
0.06	15	1.9	25	1.7	2955	505.0
0.07	12	1.5	24	2.0	2894	515.6
0.08	12	1.5	24	2.0	2894	515.6

The following Table corresponds to the application of the unitary impulse near a mid-side of domain border (Figure 34) and a threshold $S = 1.0$ has been chosen:

k	to be updated	% tot	updated	$\frac{\text{updated}}{\text{to be updated}}$	GreenLU flops	$\frac{\text{LU flops}}{\text{GreenLU flops}}$
0.03	80	10.1	137	1.7	58773	25.4
0.05	31	3.9	107	3.4	35205	42.4
0.06	25	3.2	95	3.8	32126	46.4
0.07	17	2.1	84	4.9	23125	64.5
0.08	14	1.8	72	5.1	18465	80.8

5.3 Conclusions and Further Developments

The two approaches presented in this Chapter, namely the Waves-Bordering and the GreenLU, differ mainly in the way they determine the subset of equations to be factored. The Waves-Bordering method is generally applicable but requires extra computations for the recursive updating of the solution from the previous temporary-subset to the next enlarged one. The GreenLU method determines a-priori the subset of equations to be eliminated, but requires a characterization of the *system impulse response* (i.e. the Green's function).

From the GreenLu approach it can be derived also an approximate direct solver (i.e. it can produce an incomplete factorization in which the threshold to compute or not the updates is determined again with the help of the superposition principle and the *system impulse response* characterization). This is currently under development.

The GreenLu method can also be applied to iterative methods. An interesting application would be to apply the incomplete factorization just mentioned as a pre-conditioner.

References

[1] M.Ainsworth, J.T.Oden "A unified Approach to A-Posteriori Error Estimation using Element Residual methods", *Numer. Math.* **65** (1993) 23-50

[2] M.Ainsworth, J.T.Oden "A-Posteriori Error Estimation in Finite Element Analysis", *Comp.Meth.Appl.Mech.Eng.* **142** (1997) 1-88

[3] P. Amestoy, T. Davis, I. Duff. "An Approximate Minimum Degree Ordering Algorithm", *SIAM J. Matrix Anal. Appl.* **17** (1996) n.4, 886-905

[4] O.Axelsson, V.Barker "Finite Element Solution of Boundary-Value Problems", Computer Science and Applied Mathematics, Academic Press, 1984

[5] I.Babuska, A.K.Aziz "On the Angle Condition in the Finite Element Method" *SIAM J. Numer. Anal.* **13** (1976), no.2, 214-226

[6] I.Babuska, W.C.Rheinboldt "Error Estimates for Adaptive Finite Element Computations", *SIAM J. Numer. Anal.* 15 (1978) n.4, 737-754

[7] I.Babuska, B.Szabo' "Finite Element Analysis", John Wiley & Sons, 1991

[8] I.Babuska et al. "Validation od A-Posteriori Error Estimators by Numerical Approach", *Int.J.Numer.Meth.Eng.* **37** (1994) 1073-1123

[9] I.Babuska *et al.* "Pollution-Error in the h-version of the Finite Element Method and the local Quality of A-Posteriori Error Estimators", *Finite Elements in Analysis and Design* **17** (1994) 273-321

[10] I.Babuska, G.Caloz, J.E.Osborne "Special Finite Element Methods for a Class of Second Order Elliptic Problems with Rough Coefficients" *SIAM J. Numer. Anal.* **31** (1994) n.4, 945-981

[11] I.Babuska *et al.* "A Posteriori Estimation and Adaptive Control of the Pollution Error in the h-version of the Finite Element Method" *Int.J.Numer.Meth.Eng.* **38** (1995) 4207-4235

[12] I.Babuska et al. "Pollution Error in the h-Version of the Finite Element Method and the Local Quality of the Recovered Derivatives", *Comp.Meth.Appl.Mech.Eng. 140* (1997) 1-37

[13] I.Babuska et al. "A Posteriori Error Estimation for Finite Element Solutions of Helmholtz Equation - Part II: Estimation of the Pollution Error", *Int.J.Numer.Meth.Eng.* **40** (1997) 3883-3900

[14] I.Babuska, S.Sauter "Is the Pollution Effect of the FEM avoidable for the Helmholtz Equation considering High Wave Numbers ?", *SIAM J.Numer.Anal.* **34** (1997) n.6, 2392-2423

[15] R.E.Bank "Hierarchical Bases and the Finite Element Method" *Acta Numerica* (1996) 1-43

[16] R.E.Bank "Mesh smoothing using a posteriori error estimates" *SIAM J. Numer. Anal.* **34** (1997), no. 3, 979-997

[17] R.Becker, R.Rannacher “A Feed-Back Approach to Error Control in Finite Element Methods: Basic Analysis and Examples”, *East-West J. NUmer. Math.* **4** (1996), n.4, 237-264

[18] A.Bensoussan, J.L.Lions, G.Papanicolau “Asymptotic Analysis for Periodic Structures” North-Holland, 1978

[19] G.Beylkin, R.R.Coifman, V.Rokhlin “Fast Wavelet Transforms and Numerical Algorithms I”, *Comm. Pure and Appl. Math.* **44** (1991) 141-183

[20] D. Braess *Finite Elements*, Cambridge University Press (1997)

[21] S.Brenner, L.Scott “The Mathematical Theory of Finite Element Methods”, Springer, 1994

[22] M.E.Brewster, G.Beylkin “A Multiresolution Strategy for Numerical Homogenization” *Appl. and Comput. Harmonic Analysis* **2** (1995) n.4, 327-349

[23] C.Brezinski, M.Redivo-Zaglia, H.Sadok ”A Breakdown-free Lanczos typ e Algorithm for Solving Linear Systems”, *Numerische Mathematik* 63 (1992) 29-38

[24] C.Brezinski, M.Morandi Cecchi, M.Redivo-Zaglia “The Reverse Bordering Method”, *SIAM J. Matrix Anal. Appl.* **15** (1994) n.3, 922-937

[25] F.Brezzi, L.P.Franca, T.J.R.Hughes and A.Russo ”Stabilization Techniques and Subgrid Scales Capturing”, in *The State of the Art of Numerical Analysis* (1997) 391-406

[26] F.Brezzi, L.P.Franca, T.J.R.Hughes and A.Russo ”$b = \int g$”, *Comp.Meth.Appl.Mech.Eng.* **145** (1997) 329-339

[27] G.Carey “Computational Grids”, Taylor&Francis, 1997

[28] M.Morandi Cecchi, A.Cella “An Extended Theory for the Finite Element Method”, in *Variational Methods in Engineering*, vol.1, C.A.Brebbia ed., Southampton University Press, 1973

[29] M.Morandi Cecchi ”A Study of the Behaviour of Oscillatory Waves in a Lagoon”, *Int.J.Numer.Meth.Eng.* **27** (1989) 103-112

[30] M.Morandi Cecchi, R.Nociforo ”Discrete Finite Elements Method in Space-Time Domain for Parabolic Linear Problems” *Le Matematiche* **XLVI** (1991) II, 655- 673

[31] M.Morandi Cecchi, M.Redivo Zaglia, G.Scenna ”Approximation of the Numerical Solution of Parabolic Problems” , Computational and Applied Mathematics I, C.Brezinski and U.Kulish eds. Elsevier Science Publishers, 1992 IMACS

[32] M.Morandi Cecchi, A.Scarpa ”A Space-Time Finite Element Method for Nonlinear Parabolic Problems”, *Applied Numerical Mathematics* **15** (1994) 247-258

[33] M.Morandi Cecchi, R.Nociforo, P.Patuzzo Grego ”Space-Time Finite Elements Numerical Solution of Burgers Problems” , *Le Matematiche* **LI** (1996) I, 43-57

[34] M.Morandi Cecchi,A.Pica,E.Secco "A Projection Method for Shallow Water Equations", *Int.J.Num.Meth. in Fluids* **27** (1998) 81-95.

[35] M.Morandi Cecchi, L.Salasnich "The Shallow Waters Equations: State-of-the-Art and New trends", *Computational Fluid Dynamics Review*, vol. II, M.Hafez and K.Oshima eds., 1998, 972-993

[36] M.Morandi Cecchi, F.Marcuzzi "Adaptivity in Space and Time for Shallow Water Equations", *Int.J.Numer.Meth.Fluids* **31** (1999) 285-297

[37] M.Morandi Cecchi, F. Marcuzzi "An Approximate Waves-Bordering Algorithm for Adaptive Finite Elements Analysis", *Numerical Algorithms* **21** (1999) 311-322

[38] M.Morandi Cecchi, F.Marcuzzi "A Fast Numerical Homogenization Algorithm for Finite Element Analysis", *Int. Journal of Num. Methods in Engineering* **46** (1999) 1639-1649

[39] M.Morandi Cecchi, F.Marcuzzi "Adaptivity in Space-Time Finite Element Approximations" presented at "EPMESC VII", Macao, 1999

[40] M.Morandi Cecchi, F.Marcuzzi "A linear System Solver for Adaptive FEM", presented at the *European Conference on Computational Mechanics (ECCM) '99* ", Munchen, 1999

[41] P.G.Ciarlet "Basic Error Estimates for Elliptic Problems" in *Handbook of Numerical Analysis*, P.G.Ciarlet and J.L.Lions eds., North-Holland, Amsterdam (1991), 20-343

[42] P.G.Ciarlet and J.L.Lions eds. "Handbook of Numerical Analysis: vol. II - Finite Element Methods (Part 1)", North-Holland, Amsterdam, 1991

[43] P.Clement "Approximation by Finite Element Functions using Local Regularization", *RAIRO Anal.Numer.* **2** (1975) 77-84

[44] A. Cline, C. Moler, G. Stewart, J.Wilkinson. "An Estimate for the Condition Number of a Matrix", *SIAM J. Sci. and Stat. Comput.* **16** (1979) 368-375

[45] P. Concus, G. Golub, D. O'Leary "A Generalized Conjugate Gradient Method for the Numerical Solution of Elliptic Partial Differential Equations", in *MAA Studies in Matematics* (1984) 178-198

[46] N.A.Coult "A Multi-Resolution Strategy for Homogenization of Partial Differential Equations" PhD Thesis, University of Colorado at Boulder, 1997

[47] P.Davis "Interpolation and Approximation", Dover Pubblications, 1963

[48] T.Davis, http://www.cise.ufl.edu/ davis/colamd/

[49] J.Demmel, S.Eisenstat, J.Gilbert, X.Li, J.Liu "A Supernodal Approach to Sparse Partial Pivoting", LAPACK Working Note #103, ftp.unina.it/pub/netlib/lapack/lawn

[50] J.Demmel, X.Li "SuperLU: Sparse Gaussian Elimination on High Performance Computers", http://www.cs.berkeley.edu/ demmel/SuperLU.html

[51] J.Donea “A Taylor-Galerkin Method for Convective Transport Problems” *Int. J. Num. Meth. Eng.* **20** (1984) 101-119

[52] I. Duff “Sparse Numerical Linear Algebra: Direct Methods and Preconditioning”, in *The State of the Art of Numerical Analysis* (1997) 27-62

[53] K.Eriksson, C.Johnson “Adaptive Finite Element Methods for Parabolic Problems I: A Linear Model Problem”, *SIAM J. Numer. Anal.* **28** (1991) 43-77

[54] K.Eriksson, C.Johnson “Adaptive Finite Element Methods for Parabolic Problems II: Optimal Error Estimates in $L_\infty L_2$ and $L_\infty L_\infty$”, *SIAM J. Numer. Anal.* **32** (1995) 706-740

[55] K.Eriksson,D.Estep,P.Hansbo,C.Johnson “Introduction to Adaptive Methods for Differential Equations” *Acta Numerica* (1995) 105-179

[56] K.Eriksson, C.Johnson “Adaptive Finite Element Methods for Parabolic Problems IV: Nonlinear Problems”, *SIAM J. Numer. Anal.* **32** (1995) 1729-1749

[57] K.Eriksson, C.Johnson “Adaptive Finite Element Methods for Parabolic Problems V: Long-Time Integration”, *SIAM J. Numer. Anal.* **32** (1995) 1750-1763

[58] K.Eriksson, C.Johnson “Adaptive Finite Element Methods for Parabolic Problems VI: Analytic Semi-Groups”, *SIAM J. Numer. Anal.* (1999)

[59] R.Ferencz, T.J.R.Hughes ”Iterative Finite Element Solutions in nonlinear Solid Mechanics”, in *Handbook of Numerical Analysis, vol. VI*, P.G.Ciarlet and J.L.Lions eds., Elsevier Science (1998), 2-178

[60] L.Franca, S.Frey, T.J.R.Hughes “Stabilized Finite Element Methods: I. Application to the Advective-Diffusive Model”, *Comp.Meth.Appl.Mech.Eng.* **95** (1992) 253-276

[61] D.A.French, T.E.Peterson ”A Continuous Space-Time Finite Element Method for the Wave Eqaution” *Mathematics of Computation* **65** (1996) n.214, 491-506

[62] A.George, J.W.H.Liu “A Minimal Storage Implementation of the Minimum Degree Algorithm” *SIAM J. Numer. Anal.* **17** (1980) 282-299

[63] J.Gilbert, C.Moler, R.Schreiber ”Sparse Matrices in MATLAB: Design and Implementation”, *SIAM J.Matrix.Anal.Appl.* **13** (1992) 333-356

[64] G. Golub, C.Van Loan *Matrix Computations*, third edition, John Hopkins Univ. Press (1996)

[65] G. Golub, H. van der Vorst “Closer to the Solution: Iterative Linear Solvers”, in *The State of the Art of Numerical Analysis* (1997) 63-92

[66] R. Haberman ”Elementary Applied Partial Differential Equations”, prentice-Hall, 1987

[67] E.Hairer, S.P.Norsett, G.Wanner “Solving Ordinary Differential Equations I”, Springer-Verlag, 1987

[68] P.Hansbo "Adaptivity and Streamline Diffusion Procedures in the Finite Element Method", PhD Thesis, Chalmers University of technology, Goteborg, 1989

[69] I. Harari, T.J.R. Hughes "What are C and h?: Inequalities for the analysis and design of finite elements methods", *Comp. Meth. Appl. Mech. Eng.* **97** (1992) 157-192

[70] M.T.Heath, P.Raghavan "A Cartesian parallel Nested Dissection Algorithm" *SIAM J. Matrix Anal. Appl.* **16** (1995) n.1, 235-253

[71] W.D.Henshaw "Automatic Grid Generation", *Acta Numerica* (1996) 121-148

[72] T.J.R.Hughes, Brooks "Streamline Upwind Petrov-Galerkin Formulations for Convection Dominated Flows with particular emphasis on the Incompressible Navier-Stokes Equations", *Comp.Meth.Appl.Mech.Eng.* **32** (1982) 199-259

[73] T.J.R.Hughes "The finite element method: linear static and dynamic finite elements analysis", Prentice-Hall, 1987

[74] T.J.R.Hughes, G.Hulbert "Space-Time Finite Element Methods for Elastodynamics: Formulations and Error Estimates", *Comp.Meth.Appl.Mech.Eng.* **66** (1988) 339-363

[75] G.Hulbert, T.J.R.Hughes "Space-Time Finite Element Methods for Second-Order Hyperbolic Equations", *Comp.Meth.Appl.Mech.Eng* **84** (1990) 327-348

[76] T.J.R. Hughes "Multi-Scale Phenomena: Green's Functions, the Dirichlet-to-Neumann Formulation, Subgrid Scale Models, Bubbles and the origins of Stabilized Methods", *Comp.Meth.Appl.Mech.Eng.* **127** (1995) 387-401

[77] T.J.R. Hughes, J.Stewart "A Space-Time Formulation for Multi-Scale Phenomena", *J.Comput.Appl.Math.* bf 74 (1996) 217-229

[78] T.J.R. Hughes *et al.* "The Variational multi-Scale Method - a Paradigm for Computational Mechanics", *Comp.Meth.Appl.Mech.Eng.* **166** (1998) 3-24

[79] B.M.Irons "A Frontal Solution Program for finite Element Analysis", *Int.J.Numer.Meth.Eng.* 2 (1970) 5-32

[80] C.Johnson "Numerical Solution of Partial Differential Equations by the Finite Element Method", *Cambridge University Press*, 1987

[81] C.Johnson, R.Rannacher, M.Boman "Numerics and Hydrodynamic Stability: Toward Error Control in Computational Fluid Dynamics", *SIAM J. Numer. Anal.* **32** (1995) n.4, 1058-1079

[82] C.Johnson, A,Szepessy "Adaptive Finite Element Methods for Conservation Laws Based on A-Posteriori Error Estimates", *Comm. Pure Appl. Math.* **48** (1995) 199-234

[83] H.Kardestuncer "Matrix Analysis of Discrete Elastic Systems", CISM Courses and Lectures 179, Springer 1972

[84] J.W.H.Liu "The Multifrontal Method and Paging in Sparse Cholesky Factorization" *ACM Transactions on Mathematical Software* **15** (1989) 310-325

[85] J.W.H.Liu "The Multifrontal Method for Sparse Matrix Solution: Theory and Practice" *SIAM Review* **34** (1992) n.1, 82-109

[86] D.G.Luenberger "Optimization by Vector Space Methods", Wiley, 1978

[87] A.Masud, T.J.R.Hughes "A space-time Galerkin/least-squares finite element formulation of the Navier-Stokes equations for moving domain problems", *Comp.Meth.Appl.Mech.Eng.* **146** (1997) n.1-2, 91-126

[88] S.A.Meguid, Z.H.Zhu "A Novel Finite Element for treating Inhomogeneous Solids" *International Journal for Numerical Methods in Engineering* **38** (1995) 1579-1592

[89] METIS library's Home Page, http://www-users.cs.umn.edu/ karypis/metis/

[90] P.A.Raviart, J.M.Thomas "Introduzione all'analisi numerica delle equazioni alle derivate parziali", *Collana di Matematica Applicata e Numerica*, Masson, 1989

[91] R.Rodriguez "Some Remarks on Zienkiewicz-Zhu Estimator", *Num. Meth. for PDE* **10** (1994), n.5, 625-635

[92] J.Ruppert "A Delaunay Refinement Algorithm for Quality 2-dimensional Mesh Generation" *Journal of Algorithms* **18** (1995) n.3, 548-585

[93] J.R.Schewchuk "Triangle: A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator" at http://www.cs.cmu.edu/∼quake/triangle.html

[94] B.Smith, P.Bjorstad, W.Gropp "Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations", *Cambridge University Press*, 1996

[95] I. Stakgold. *Green's Functions and Boundary Value Problems*, Academic Press (1983)

[96] J.Stewart, T.J.R.Hughes "A Tutorial in Elementary Finite Element Error Analysis: a Systematic Presentation of A-Priori and A-Posteriori Error Estimates", *Comp.Meth.Appl.Mech.Eng.* **158** (1998) 1-22

[97] G.Strang, G.Fix "An Analysis of the Finite Element Method", Prentice-Hall, 1973

[98] V.Thomee "Finite Difference Methods for Linear Parabolic Equations" in *Handbook of Numerical Analysis*, P.G.Ciarlet and J.L.Lions eds., North-Holland, Amsterdam (1991) 12-171

[99] R.Verfurth "A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques", *J.Comp.Appl.Math.* **50** (1994) 67-83

[100] R.Verfurth "A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques" Wiley-Teubner, 1997

[101] L.B.Wahlbin "Local Behavior in Finite Element Methods" in *Handbook of Numerical Analysis*, P.G.Ciarlet and J.L.Lions eds., North-Holland, Amsterdam (1991), 353-522

[102] H.Yserentant "On the Multi-Level Splitting of Finite Element Spaces", *Numer. Math.* **49** (1986) 379-412.

[103] L.Zeng "On Adaptive Finite Element Procedures for Static and Dynamic Problems", PhD Thesis, Chalmers University of Technology, Goteborg, 1991

[104] Z.Zhang, H.D.Victory "Mathematical Analysis of Zienkiewicz-Zhu's Derivative Patch Recovery Technique", *Numer.Meth. for PDEs* **12** (1996) 507-524

[105] O.Zienkiewicz, K.Morgan "Finite Element Approximations", Wiley, 1984

[106] O.C.Zienkiewicz and J.Z.Zhu "A Simple Error Estimator and Adaptive Procedure for Practical Engineering Analysis" *Int. J. Num. Meth. Eng.* **24** (1987) 337-357

[107] O.C.Zienkiewicz, and C.Taylor "The Finite Element Method" fourth edition, McGraw-Hill, 1989

[108] O.C.Zienkiewicz, J.Z.Zhu "The Superconvergent Patch Recovery and A-Posteriori Estimates", Part I and II, *Int.J.Numer.Meth.Eng.* **33** (1992) 1331-1382

[109] O.C.Zienkiewicz,J.Wu,J.Peraire "A New Semi-Implicit or Explicit Algorithm for Shallow Water Equations" *Math. Modell. and Sci. Comput.* **1** (1993) 31-49

[110] O.C.Zienkiewicz, J.Wu "Automatic Directional Refinement in Adaptive Analysis of Compressible Flows", *Int.J.Numer.Meth.Eng.* **37** (1994) 2189-2210