McGill University
School of Computer Science
ACAPS Laboratory

Advanced Compilers, Architectures and Parallel Systems

Deepness Analysis: Bringing Optimal Fronts to Triangular Finite Element Method

Jérôme Galtier

ACAPS Technical Memo 73
September 30, 1994

galtier@clipper.ens.fr

Stage de DEA sous la direction de Laurie Hendren

ACAPS • School of Computer Science • 3480 University St. • Montréal • Canada • H3A 2A7

Abstract

A scheme is presented for analising finite-element triangulations. The method takes a random triangulated planar graph and gives a multifrontal way to solve the corresponding physical problem, so that the maximum bandwidth of each front is guaranteed to be "optimal". The interesting characteristic of this scheme is that it introduces large-grained parallelism dictated by the domain structure.

A way to extend these results to unsymmetric systems is given. Some experimental tests are also presented.

Contents

List of Figures

1 Introduction

In this paper, we focus on the direct solution of linear systems of equations

$$\mathrm{A.x} = \mathrm{b}$$

where the matrix A has the following property: there exists a triangulated planar graph $G = (V, E)$ where $V = (v_1, \ldots, v_n)$ satisfying:

$$\forall i, j \in \{1, \ldots, n\} \quad i \neq j \text{ and } A_{i,j} \neq 0 \Rightarrow v_i \text{ and } v_j \text{ are adjacent}$$

This property characterises a very large class of matrices, and more particularly those arising from the finite-element method in two dimensions. In fact, the idea of triangulated planar graphs - also refereed to as *triangulations* of a physical domain Ω - emphases the discretisation of a physical problem. Another issue for that kind of problems is the use of iterative schemes [7, 12].

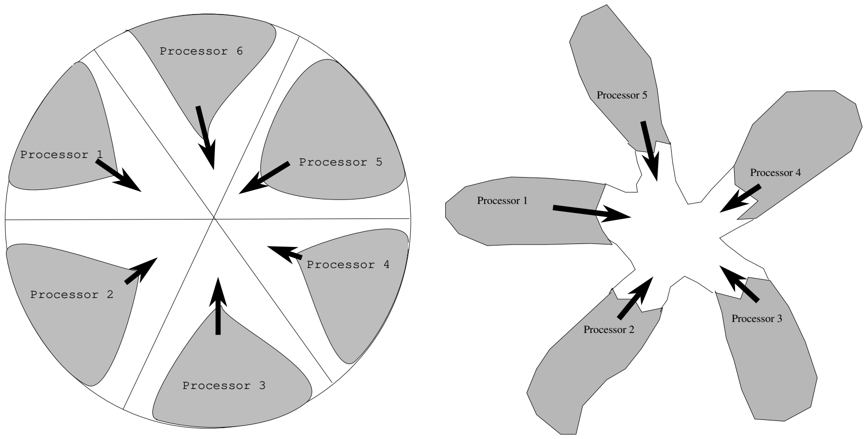
Direct methods are generally adaptations of the gaussian elimination to sparse matrices. In order to minimise the computational time of a resolution, people have introduced the *fill-in* parameter [16]. It is well known that minimising the fill-in on a given sparse matrix is an NP-complete problem [19]. However, many heuristics have been developed so far.

For instance, Irons [11] and Hood [10] described the frontal method in the early seventies. Then George [8] discovered a way to reduce significantly the fill-in on the square grid, and Hoffman and al. [9] could prove that this result cannot be improved by an order of magnitude. Later, those schemes were extended to many other classes of graph, and more particularly planar graphs [14].

Moreover, for physical problems, it is important to have a slightly different view on this sequential computational approaches. For instance, vectorisation allows us to do many operations at the same time if the problem is regular enough. The distributed parallelism that one can give to a particular calculation will also greatly improve timings for a wide class of architectures.

It is also interesting to link this recent concept of computation with mathematical properties that characterise the sparsity structure of the matrix.

In particular, we would like to automatically detect some main properties which are to be exploited to draw some multifrontal schemes. Figure 1 gives some interesting cases to exploit that are not detected - to our knowledge - by classical analysis.



Cake-shaped domain　　　Star-shaped domain

Figure 1: Some interesting cases to exploit

1.1 Subject Concerned

Here we do not take the whole subject of fronts into consideration, but we restrict ourselves to *triangulated planar graphs.* Even if this restriction is quite important, we stress the fact that triangles are a very popular structure in Numerical Analysis for their simple characteristics. In particular, bijections from a random triangle to the standard one are naturally linear, so that errors are less important and computation is faster.

More practically, a graph - or mesh - will be a pair $G = (X, E)$, where X is the set of *vertices* or *nodes*, and $E \subseteq \{\{x, y\} \in X^2,\ x \neq y\}$ is the set of edges. Two nodes x, y such that $\{x, y\} \in E$ are said to be *adjacent.* The graph is bi-dimensional or planar in the sense that it may be drawn on a map without crossing edges. We are concerned here with simply-connected triangulated domains, that is to say that the mesh is issued from the triangulation $\mathcal{T}$ of a simply connected domain Ω of $\mathbb{R}^2$. Then we distinguish a particular subset B of X, consisting of vertices of the boundary (Ω simply connected iff B connected). For a given triangle T, we note T_v the set of vertices of T, and T_e the set of edges. Finally, for $V \subset X$, we note $< V >$ the graph induced by V in G.

In section 2, we introduce a new kind of analysis particular to these triangulated planar graphs. From this we derive of section 3 a way to characterise the global shape of the graph with local properties. We also show that this information can be directly used to draw an efficient ordering in order to achieve a sparse cholevski factorisation. Then in section 4 we give some ideas to extend the results to matrices that are not necessarily definite positive. Finally, in section 5, we present some results that illustrate the practical use of our method.

2 Deepness Analysis

2.1 Basic Idea

We saw precedently that "cigar-shaped" domains were easy to analyse. Lead on straight from this idea we remarked that even if King's algorithm could not detect it, "star-shaped" domains had some latent properties exploitable by multi-frontal techniques. *So the thinner the domain, the easier elimination by bands or fronts.*

How to detect the thinness - or inversely the deepness - of a graph ? The basic idea we introduce here lies on distance to the boundary. It is obvious that detecting distance and more particularly vertices with maximal distance to the boundary, will give us an idea of what we are looking for.

Computing this information is not expensive. The following basic algorithm (linear with the number of edges) has been chosen :

- Initialisation : mark each vertex on the boundary with colour 1.
- Step n : For each vertex of colour n, check if its adjacent vertices have a colour. If not, mark them with colour $n+1$.

We have then $\forall x \in X \quad colour(x) = \delta(x, B) + 1$. We will also note later

$$\forall T \in \mathcal{T} \qquad \rfloor\wr\updownarrow\wr\sqcap\nabla(\mathcal{T}) = \sum_{\sqsubseteq \in \mathcal{T}_{\sqsubseteq}} \rfloor\wr\updownarrow\wr\sqcap\nabla(\sqsubseteq)$$

2.2 Optimal Fronts

The notion of fronts Irons introduced was very general. This was a "line of computation" on the domain which was allowed to shut on itself to become circular and then involved no more vertices on the boundary. However, we shall stick to the idea of a front which starts and ends on vertices of the boundary.

Definition 1 *An optimal front is a couple of series of vertices*

$((u_i)_{1\leq i\leq n}, (v_i)_{1\leq i\leq n})$ *with same length n verifying :*

$$\begin{array}{ll} (i) & \forall i, 1 \leq i \leq n \ u_i \text{ and } v_i \text{ have same colour } i \\ (ii) & \forall i, 1 \leq i \leq n-1 \quad \begin{cases} u_i \text{ and } u_{i+1} \text{ are adjacent} \\ v_i \text{ and } v_{i+1} \text{ are adjacent} \end{cases} \\ (iii) & \text{Either } u_n = v_n \text{ or } u_n \text{ and } v_n \text{ are adjacent} \\ (iv) & \text{If } n > 1, \text{ then } u_n \neq v_n \text{ or } u_{n-1} \neq v_{n-1} \end{array}$$

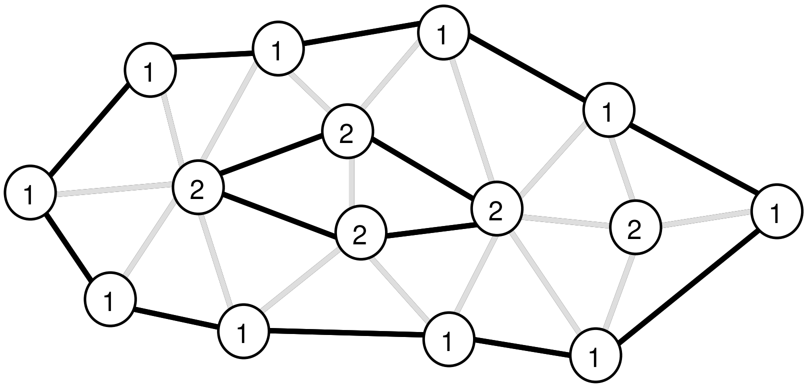


Figure 2: Stong connectivity lines

Properties (ii) and (iii) guarantee front cohesion. Property (i) means optimality, which is closely linked to the actual representation. Finally, property (iv) avoids empty fronts.

The following property is straightforward.

Property 1 *A front which touches the boundary, contains u_n and v_n, and holds two distinct arms has at least as many vertices as an optimal front.*

We can see that considering two ordered arms ($(u_i, v_i) \neq (v_i, u_i)$) allows us to define an orientation. This way, we can call one arm "right" and the other "left", and "inside" or "outside" the front becomes meaningful regardless of the assembly state - provided the fact the arms do not "cross" each other (i.e. don't have common nodes). In this context we do not need notions like macro- or generalised element anymore, for it is replaced in terms of optimal fronts.

But is it possible to restrict computational fronts to optimal fronts ? In other terms, can we move these fronts step by step along the graph ?

We shall see that in case it is possible, nodes of a same colour cover some lines named lines of strong connectivity.

2.3 Strong Connectivity

Definition 2 *We call strong connected component C_i^j of a given colour i, each connected component in the graph $G_i = (X_i, E_i)$ with :*

$$X_i = \{x \in X; colour(x) = i\}$$
$$E_i = \{e \in E;\ \exists T \in \mathcal{T};\ \rfloor\wr\updownarrow\wr\sqcap\nabla(\mathcal{T}) \geq \ni\rangle\ \dashv\backslash\lceil\ \rceil \in \mathcal{T}_\rceil\}$$

The set V_i emphasizes the "elimination" of all triangles having a vertex of colour $< i$, which lets some "islands" appear. Hence the following notion :

Definition 3 *We call local boundary of one strong connected component C_i^j of a given colour i the set $\mathcal{B}_{\rangle}^{|}$ of edges such that, if $C_i^j = (C_V^{i,j}, C_E^{i,j})$*

$$\begin{aligned}\mathcal{B}_{\rangle}^{|} =< \mathcal{C}_{\mathcal{E}}^{\rangle,|} > \cap \{ \quad & e \in E;\ \exists T_1, T_2 \in \mathcal{T};\ \rceil \in \mathcal{T}_{\infty\rceil} \cap \mathcal{T}_{\in\rceil} \\ & \textit{and } colour(T_1) = 3i - 1 \textit{ and } colour(T_2) \geq 3i\ \}\end{aligned}$$

The local boundary $\mathcal{B}_{\rangle}$ is in fact the set of edges which forms the boundary of the remaining domain after the "elimination" of triangles having a vertex of colour $< i$. An edge belonging to one $\mathcal{B}_{\rangle}$ will later be called a *strong edge.*

Property 2 *If Ω is simply connected, the path between two vertices of a same strong component may be drawn on strong edges (i.e. on the local boundary).*

This property is easy to feel. Proving it seems to be harder. The proof given here is quite long but is not vital for understanding next sections.

We start with an intermediate property.

Lemma 1 *Each vertex involves an even number of strong edges.*

Proof Let us consider the set of adjacent nodes to one vertex M (noted $Adj(M)$) with the associated edges. We try to show that each strong edge may be exactly coupled to another. For this, we write $Adj(M) = \{N_0, \ldots, N_{p-1}\}$ so that $\{M, N_i, N_{i+1}\}$ is a triangle for all i in $\{0, \ldots, p-2\}$ and for $i \geq p$, we note N_i the vertex N_{imodp}.

We can then define two classes of N_i such that $\{M, N_i\}$ is a strong edge. We define $k = colour(M)$. Now N_i is in the first class iff $colour(\{M, N_i, N_{i+1}\}) \geq 3k$. The second class is formed by the N_i such that $\{M, N_i\}$ is strong and $colour(\{M, N_i, N_{i+1}\}) = 3k - 1$. These classes are distinct by definition of strong edges. We shall show that there is a bijection from the first class to the second one.

let us take N_i in the first class. N_{i+1} has a colour greater than or equal to k, and we consider the maximum number q so that vertices $N_j, i+1 \leq j \leq q$ have a colour $\geq n$. Since M is adjacent to at least one vertex of colour $k-1$, q exists and $\{N_i\}_{i+1 \leq i \leq q}$ does not include N_i. So N_q is such that N_{q+1} has colour $k-1$. Then N_q has colour k, the associated edge $\{M, N_q\}$ is strong and N_q belongs to the second class previously defined. It is obvious that if we take two distinct vertices N_i and $N_{i'}$ of the first class with $i < i' < i+p$, then we shall have $colour(N_{i'-1}) = k-1$ and $colour(N_{i+p-1}) = k-1$, so that $q < i' < q' < i+p$ for the

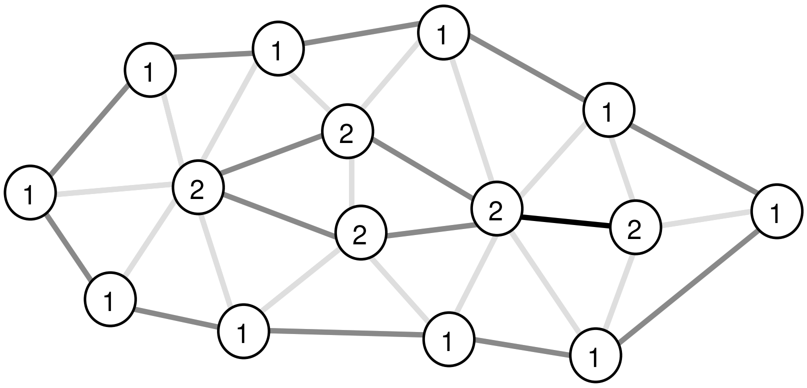


Figure 3: A weak edge

corresponding associated vertices. We have built an injection from the first class to the second, and we leave to the reader the proof that this function has an inverse. □

Proof [of Property 2]

Given this basic property we choose a vertex M in the strong component. Let us consider all its "strong" neighbours (i.e. adjacent to M by a strong edge), and the strong neighbours of them, etc... Since each node among those belongs to the strong component and has a pair number of neighbours, we have formed a chain on the domain Ω. If Ω is simply connected, inside this frontier, nodes have a colour $> n$, and so do not belong to the component. Outside, vertices cannot belong to the component. So all nodes of the component belong to the chain. □

We saw that edges with a given connected component draw level lines on the mesh. That is why it is named strong connectivity.

Among a same colour, some strong components are closer than others. They form what we call weak components.

2.4 Weak Connectivity

Definition 4 *We call weak connected component of a given colour i each connected component in the graph $< X_i >$ induced by the vertices of colour i.*

The following property follows :

Property 3 *Weak connected components are over-sets of strong connected ones.*

In the next paragraphs we shall call *weak edge* each edge linking two different strong connected components. Another interesting property concerns the general structure of typically weak edges :

Property 4 *If Ω is simply connected, the graph that weak edges draw on strong components is acyclic.*

Proof

Strong components of the colour i have two distinct weak edges. Since Ω is simply connected, inside each strong component we may draw a path with vertices of colour i among the possible two vertices concerned. Hence the cycle only made with vertices of the colour i :

$$\begin{array}{lccc} \text{Component 1} & A & \leftrightarrow & B \\ \text{Weak edges} & \updownarrow & & \updownarrow \\ \text{Component 2} & C & \leftrightarrow & D \end{array}$$

Using one more time simple connectivity of Ω, we see that inside the cycle, vertices have the colour $\geq i+1$, and so our old weak edges are not really weak.

□

2.4.1 Very-strong connectivity

The structure of strong components is sometimes more complicated than single cycles. The notion of very-strong connectivity is meant to make one understand this more efficiently. The interest of very-strong connectivity is concentrated on vertices with more than two strong edges. These introduce some additional cycles among a single strong component. We have the following statements :

Definition 5 *A multiple vertex is a vertex with at least four strong edges.*

The reader will prove the following:

Property 5 *The extra loops introduced by multiple vertices have globally a structure of acyclic graph - provided Ω is simply connected.*

Definition 6 *We call very-strong connected component of a strong component, the set of all triangles included in one of its "loops".*

We now have a better grasp of the inheritance among vertices with successive colours.

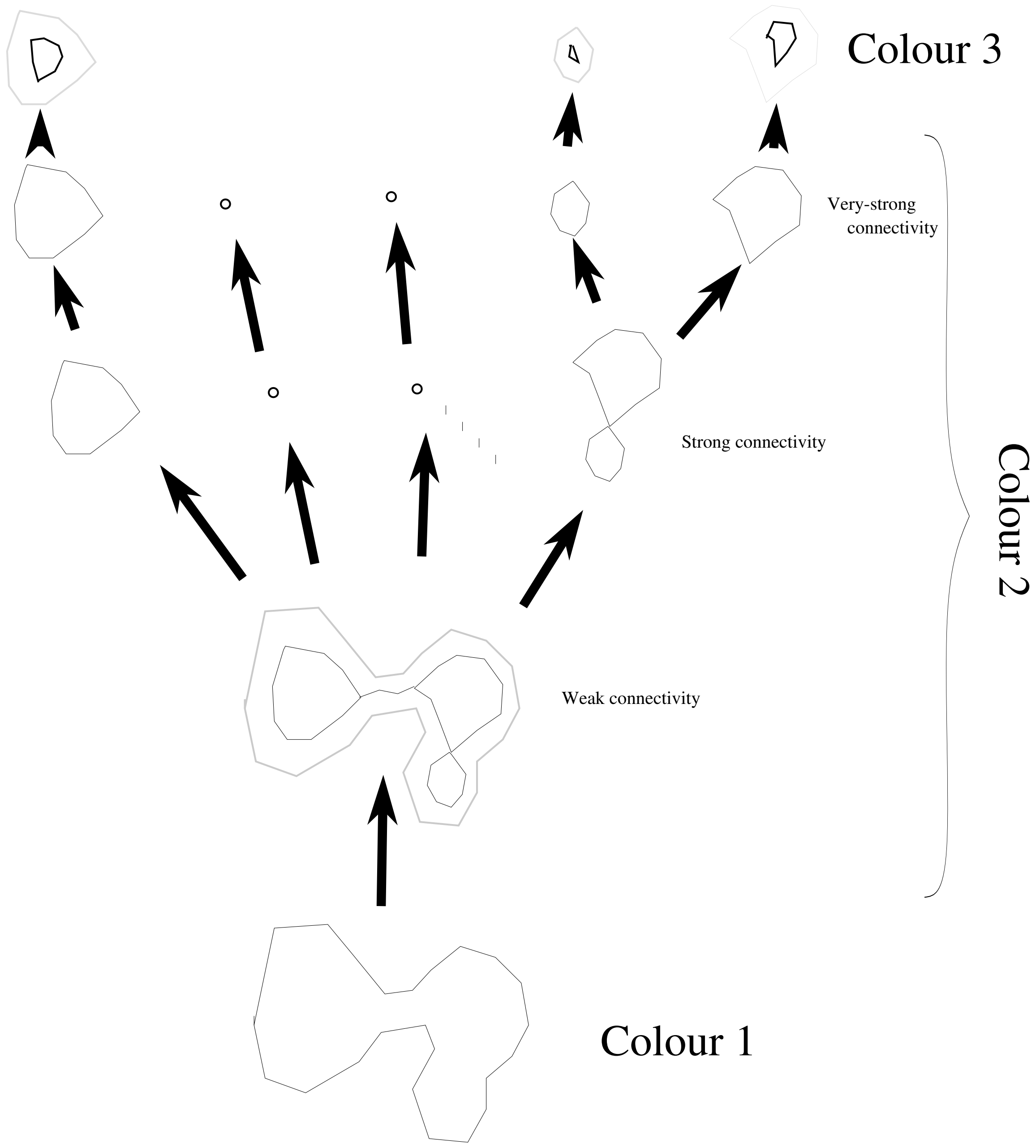


Figure 4: Connectivity hierarchy

3 Focus On Irregularity - Triangulation Spine

3.1 Regularity and Irregularity

How to define what a regular triangulation is ? According to previous notions, we would like to say that it is a triangulation which can be described only by optimal fronts. Of course,

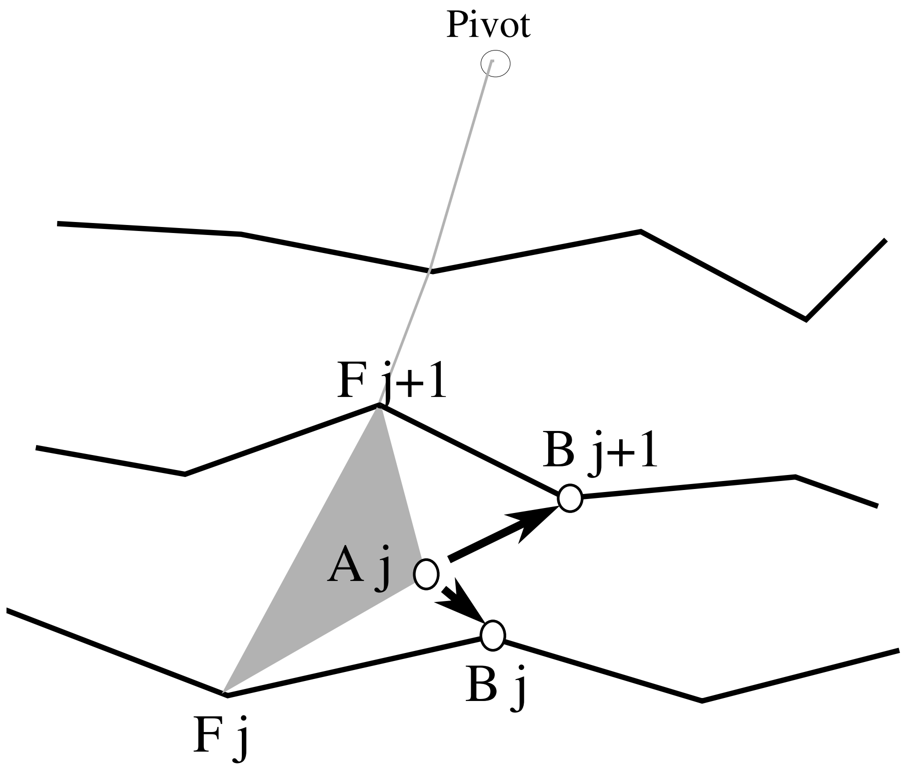


Figure 5: Definition of regularity

description means step-by-step improvement, where a step involves a few number of elements, minimal if possible.

What is this minimal number ? If we modify the front one vertex after the other, we shall involve two elements each time - except in the case of vertices on the boundary. So we shall try to keep this minimal case.

Structures leading to irregularity are not so common. The previous chapter showed some of them. The following definition will be useful.

Definition 7 *The set of internal edges is defined ofr each colour i by*

$$I_i = E_i - \bigcup_j \mathcal{B}_i^j$$

An internal edge is also an edge between two nodes of the same colour which is neither strong nor weak.

The general case which occurs before an optimal front improvement can be represented as figure 5 shows. In this figure, dark edges represent current strong edges (“current” means here belonging to the very-strong component where the pivot is).

To give a formal definition of regularity, we need to get in a few more details of the triangulation structure. Since the adjacency graph of our matrix is a triangulated planar graph, let us take such a practical drawing without crossing edges of our graph. Now we can describe any triangle according to a clockwise ordering. Then $T(A,B,C) = T(C,A,B) = T(B,C,A)$ but is not equal to $T(A,C,B)$ whenever both of them exit. Now since G is triangulated, for any edge $\{A,B\}$ that is not on the boundary, there exists one and only one vertex C such that $T(A,B,C)$ is an existing triangle. We shall note C $\mathbf{Next}(A,B)$.

Definition 8 *For each virtual optimal front $(F_i)_{1\leq i\leq p}$ and each pair of successive nodes F_j and F_{j+1} on it ($1 \leq j \leq p-2$), we consider the vertex $A_j = \mathbf{Next}(F_j, F_{j+1})$.*

Now, given F_k, $1 \leq k \leq p-1$, there exists a unique very-strong component $C_k^{\alpha,\beta}$ such that both F_k and F_{k+1} belong to $C_k^{\alpha,\beta}$. Now, the boundary $\mathcal{B}_k^{\alpha,\beta}$ is a cycle, and there exists a unique vertex B_k such that $\{F_k, B_k\} \in \mathcal{B}_k^{\alpha,\beta}$ and $colour(T(F_k, B_k, Next(F_k, B_k))) \geq 3i$.

The vertex A_j is in a regular position if it can be identified either to B_j or to B_{j+1}.

In other words, if we can find an A_j which does not satisfy this property, we have found some irregularity. Then, we can distinguish two cases (recall that $Colour(F_j) = j$).

I *$Colour(A_j) = j$, and since $A_j \in C_j^{\alpha,\beta}$ (We take the previous notations; α and β depend on F_j and F_{j+1}), $\{A_j, F_j\}$ is an "internal" edge.*

II *$Colour(A_j) = j+1$ and then we distinguish three other cases :*

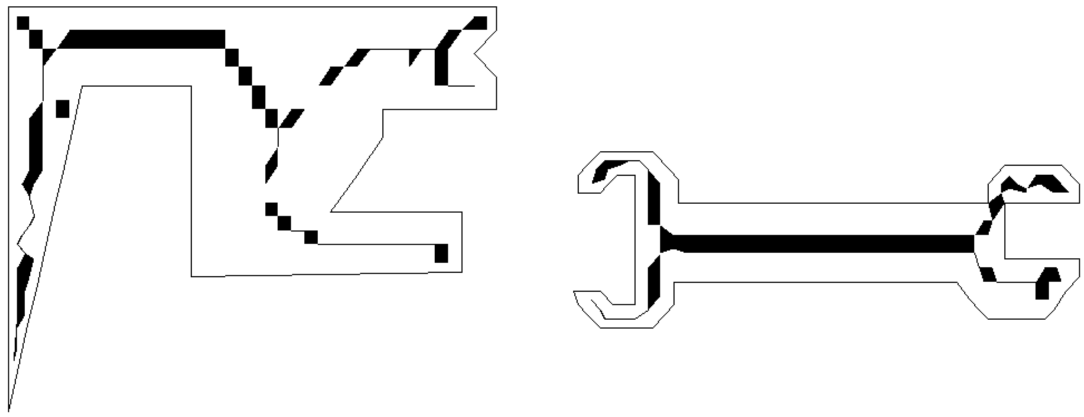
- α *$\{A_j, F_{j+1}\}$ is a "strong" edge, i.e. F_{j+1} is a multiple vertex*
- β *$\{A_j, F_{j+1}\}$ is a "weak" edge.*
- γ *$\{A_j, F_{j+1}\}$ is an "internal" edge.*

This last case needs further explanation. Turning around F_{j+1}, we see that if one of the vertices preceding or following A_j has the colour j, then A_j forms a strong edge with F_{j+1}! Since this is not the case, both A_j and F_{j+1} belongs to a very-strong connected component $C_{j+1}^{\alpha,\gamma}$ which is obviously not $C_{j+1}^{\alpha,\beta}$ (the one of F_{j+1} and F_{j+2}). This means that F_{j+1} is a multiple vertex.

As a consequence, we can check irregularities by locating the following singularities :

a	–	Internal edges
b	–	Multiple vertices
c	–	Weak edges

However, detecting irregularities independently does not really lead to some exploitable data. We also have to show how these irregularities are connected, and eventually find a means to get round them. In the next section we shall see how these structures can naturally draw some "spines" on the graph.



Part A Part B

Figure 6: Natural spine on some meshes ...

3.2 Union of Natural Irregularities : Spectacular Examples

The real problem that "irregularity" checking raised at once was mainly concentrated on the internal edges' structure. Indeed, multiple nodes are quite rare and finally quite simple : they define a kind of crossroads between very-strong components. And even weak edges are nothing more than connexions between strong components. On the contrary, internal edges can draw some very complex structures inside each very-strong component. And more precisely, they define the subdivision between very-strong components of colour n and weak components of the colour $n+1$. The problem is that an internal edge never points to what it connects - or disconnects.

So one idea consists in locating not only internal edges, but triangles of which they form an edge. In the following pictures, we checked these structures :

a Triangles having an "internal" edge

b Multiple nodes

c Weak edges

d Triangles forming a complete very-strong component

We add the last category since these triangles are local summits and consequently have an obvious interest for optimal front's way. Indeed, more mathematically, we may consider that the pivot also is a case of irregularity.

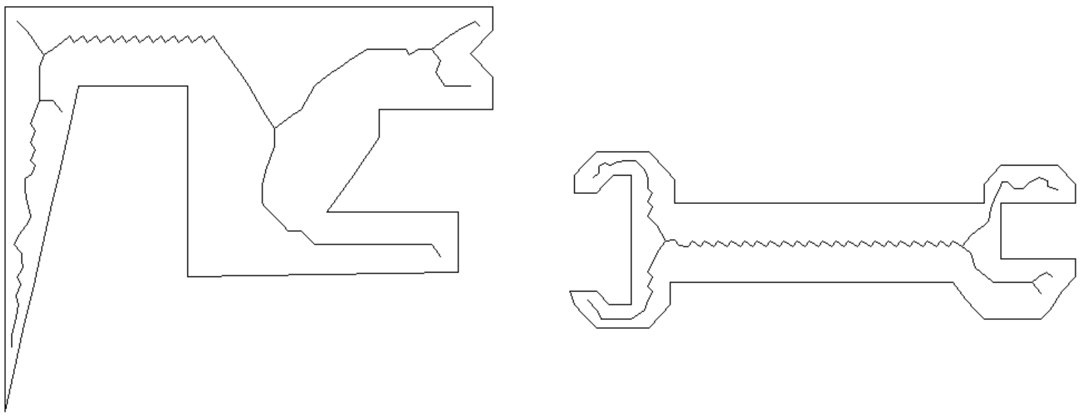


Figure 7: Machine-tree for previous examples

Let us take a few examples and see what is going on. In figure 6, part A, irregularities define a long line along the structure. On the right we can see a branch dividing itself into two parts. On the left, we note a square, which is one unexpected irregularity. In part B, irregularities are naturally connected one to another. Perhaps there is an unexpected branch down on the right. To conclude, we can say that irregularities form a very interesting set to study, in particular with the aim of describing globally the structure.

3.3 Building The Machine-Tree - Related Properties

Having an idea of what a triangulation's spine is is one thing. But making a computer able to understand this is not always very easy. Some apparently simple or very rare problems can take on a great importance on irregular graphs.

As the reader may have already noticed, the natural spine is not always naturally connected. So we have to insure this connexion while keeping fundamental properties. Another task consists in connecting intuitive structures correctly. So we propose a set of rules having those properties. For this, we will use three kinds of connexions :

(i) Between two triangles having a common edge

(ii) Between two adjacent nodes

(iii) One triangle may be connected to one of its vertices

Then we will apply the following rules (the colour of one triangle is the sum of the colours of its three vertices) :

1 *Two triangles sharing an internal edge are connected (connexion i).*

2 *The two ends of a weak edge are connected (connexion ii).*

3 *Each triangle of the colour $3i + 1$ touching an internal edge is connected to its vertex having the colour $i + 1$ (connexion iii).*

4 *The following vertices are named "departure vertices" :*

- *Multiple vertices*
- *Vertices which are ends of a weak edge*
- *Vertices with the colour $i+1$ belonging to a triangle of colour $3i+1$ which touches an internal edge*

5 *From each departure vertex, we draw a string of vertices having (strictly) increasing colour. (For a multiple vertex, we do so for each very-strong component that it is linked to, as far it is possible : if the multiple vertex has the colour n, it does not imply that we shall find in each very-strong component one adjacent node of its with the colour $n + 1$.) The process ends as soon as we join a vertex belonging to another string, or one summit of an irregular triangle. Successive vertices of the string are connected one to another (connexion ii), and the final vertex will be connected to one and only one irregular triangle it touches (connexion iii), if no string fusion has occurred.*

We notice that many of those choices are not unique. We could imagine some schemes where those artificial connexions were a little bit more guided. However, for the time being we consider this approach as a minor improvement.

Theorem 1 *If Ω is a simply-connected domain, the machine-tree has a structure of acyclic graph.*

Proof

The next sections give a proof of this theorem by decomposition on the three kinds of connectivity we saw above. We shall reason by induction on the maximal colour c_m of the triangulation.

3.3.1 Inside very-strong components

First of all let us show that the structure is completely connected, and more precisely in this part, that internal edges draw some paths to weak components of the next colour.

For this, we can consider very-strong component having more than three vertices. (Otherwise, there is no vertex inside with a superior colour, since the component is reduced to one triangle ! Remember that we decided to note these "local summits".)

Such a very-strong component is delimited by a real cycle of vertices (ie without a junction vertex). We shall then go along this cycle one vertex after the other vertex, and focus on the edges which these vertices touch inside the component.

So let us take three successive vertices A, B, and C of this external cycle. If we consider a triangulation order, we can designate some vertices M_i^A $1 \leq i \leq n_A$, as the successive adjacent nodes of A strictly between A's two strong edges (If A is a multiple vertex, all strong edges outside our very-strong component are just ignored) - so that $M_{n_A+1}^A$ should be B. We do the same for B and C; M_i^B $1 \leq i \leq n_B$ and M_i^C $1 \leq i \leq n_C$ are ordered in the same way and, logically, by extension, $M_0^B = A$, $M_0^C = B$, $M_{n_B+1}^B = C$.

Since A and B have a common edge, they share an "internal" triangle with the last vertex $M_{n_A}^A = M_1^B$. Likewise, we have $M_{n_B}^B = M_1^C$.

The edges eliminated here are either internal edges (if they are directed to a vertex of the same colour) or ascending edges (directed to a vertex of the increased colour). Let us show that the edges from B to successive vertices M_i^B and M_{i+1}^B are connected. By eliminating M_i^B and M_{i+1}^B, we can distinguish three cases :

- Their colours are superior to B's one. Since those vertices are *successive*, they are adjacent by a weak or strong edge, so that their connectivity results from induction.
- Their colours are both equal to B's one. Rule 1 insures that they are connected.
- One of them, for instance M_i^B, has the same colour as B, and the other, M_{i+1}^B, has a superior colour to B. The triangle T made by the three vertices puts the internal edge from B to M_i^B in concrete form, and is connected to M_{i+1}^B by rule 3. Then Rule 4 insures that M_{i+1}^B is considered as a departure vertex, so that by rule 5 it is connected to the actual weak component.

So we proved that "everything touching B" is connected. Naturally the same thing is true with A and C. Now we would like to show that there is a connexion from A's environment to B's one. We can consider two cases :

- M_1^B has the same colour as A and B. So rule 1 insures the connexion.
- M_1^B has a colour greater than A and B. In other words, A and B are "touching" the same weak component, hence the connexion.

Then global connectivity is insured step by step along the cycle. In fact, we have considered some internal edges twice. We also note that if a vertex happens to be a departure vertex, it will be either in contact with an internal triangle, or able to be prolonged in a superior colour.

3.3.2 Inside weak and strong components

We saw above that a strong component was the union of some very-strong components linked by some multiple vertices. Rules 4 and 5 insure that all these components will be effectively connected one to another. Property 5 shows that this spine is acyclic.

Likewise, a weak component is the union of some strong components linked by weak edges. Rules 2, 4 and 5 allow these structures to be connected.

3.3.3 Conclusion

So we have the proof by induction. We notice the importance of "artificial" rules like 4 and 5. In fact, they insure global consistency. But more practically it seems to be preferable to choose some strategies to limit those additional connexions.

Apart from those irregularities, the graph is very regular, so that optimal fronts can "move" freely. So a natural idea is to choose this irregularity spine to guide fronts in a parallel computation : irregularities are taken in consideration on the top of the front, while the two arms are guarantied to be optimal.

Of course, if the graph is very regular, the irregularity spine will have lots of useless branches, and many generalised elements will be created and then merged into others. George discovered that excessive use of nested dissections introduces a great latency in computations. We shall see in the next chapter that for non-elliptic problems, this may cause an heavy additional cost, which is logical.

3.4 An application ...

Theorem 2 *Under the following restrictions,*

a Definite-positive matrix

b Simply-connected domain Ω

c Uniformly regular triangulation

the convergence of this scheme is in $O(n^2)$*, where* n *is the number of triangles.*

In fact, this result reflects more the hard irregular case, since the convergence is dominated by front fusions. We also note that using the algorithm described in the following paragraph and additional properties due to uniform regularity, we will be able to remove the statement "a".

4 Local Pivot and Dual Elimination

4.1 The Problem

It is not always possible to insure that a matrix built from finite element is symmetric positive. This is why introducing numerical pivot is an interesting challenge. Of course, Hood [10] described a way to introduce pivots on local matrices especially adapted to the general concept of the frontal method. However, this is important to keep some control on the frontal matrix size and the previous article assumes that this size may increase arbitrarily (The solution given to excessive size of matrix is to change the maximal size parameter and rerun the programme).

So choosing for a given column a pivot row which may not be diagonally associated brings in a difficult alternative : eliminating with an incomplete row or assembly some unwanted elements.

Another significant problem - to which we have no complete solution yet - is about the double elimination of rows. Imagine that the computation is such that two fronts are to fusion. On the one side variable i has been eliminated with "frontal" row k. But on the other side, row k has already been selected to eliminate another variable j. Of course, we have to avoid that for algebraic consistency.

Why wonder about those typically fusion-associated problems ? Normally the assymptotical rate of fusion time should tend to zero, but it is however important for realistic domains. We have to carry on these problems which are doubtlessly the bottleneck of *Deepness Analysis*.

4.2 A Naive Solution : The Double Front

A first answer to the problem raised by local pivot consists in a double front. This way, during the elimination of variable i, not only will $A_{i,j}$ coefficients be given but also $A_{j,k}$ ones, where j designates any variable which would have been active in the single front scheme. So, are included in the frontal matrix, not only 1) elliptic variables (ie corresponding to a single front) but also 2) variables having non-zero coefficients with the former ones. We can then deal with pivot at any stage.

This approach raises several problems. First of all, optimality properties are lost, since we are not able to evaluate exactly the frontal matrix size. Then, in a general case, we shall double the number of variables, and so multiply computational time by four. Finally, fusion costs will have to handle much more loss. We shall have three times more variables on average , and so a computation nine times longer !

Of course, this first approach is perhaps closer to band-like methods. More practically, we could imagine a computation where pivot-rows are assembled as soon as they are needed.

And this seems to be more adapted than the previous method, but it suffers also from an inconvenient. Since assembly becomes dynamic, we cannot parallelise any more assembly and resolution. And finally, there is still no warranty on the size of frontal matrix.

4.3 A Frontal Approach : Single Front With Postponed Elimination

4.3.1 Principle

We present here an alternative method to Hood's one which uses only partial pivots but (minimises and) warranties the maximum size of the frontal matrix. This algorithm consists in considering elimination like a data-consumption : we give in entrance successive coefficients resulting from assembly, and variable elimination commands - in the same order as the elliptic case. The processor in charge of elimination has to handle these commands and manage its frontal matrix.

So, how to perform the pivot ? In fact, while starting the computation, our processor knows not only the column (i.e. the variable number) but it may also compute the row which it should use for elimination (maximum absolute value). So it is able to do the pivot choice, but generally not to achieve the elimination.

A good way to avoid the problem consists in introducing *symbolic* or *virtual* elimination : instead of forcing elimination completion, the main data to achieve it in the future is stored. For this, we introduce a *list of incomplete eliminations*. Whenever an unachievable elimination occurs, a new element is added to the list. Typically we are going to store pivot columns.

Meanwhile, it is obvious that with such a list, a frontal matrix is never up-to-date. Some operations will be needed to recover an accurate form at some points. Moreover, some additional data will be necessary : if elimination on incomplete row j has been decided, we will have to store somewhere incomplete coefficients of the row. One idea consists in storing them at the end of the frontal matrix. This additional data is called *array for row completion*.

To conclude, we can put all these structures in a unique array called *extended frontal matrix*. Data will be here stored in different areas, depending on how we use it. We remember that each row corresponds to a variable number. Let us classify those variables according to their current state; this way we obtain six categories (See figure 8).

Variables	N.E.[a]	V.E.[b]	E.[c]
	Act.	*Pass.*	*Pass.*
non-eliminated row	A	B	C
virtually eliminated row	D	$\times$	$\times$
eliminated row	$\times$	E	F

[a] non-eliminated
[b] virtually eliminated
[c] eliminated

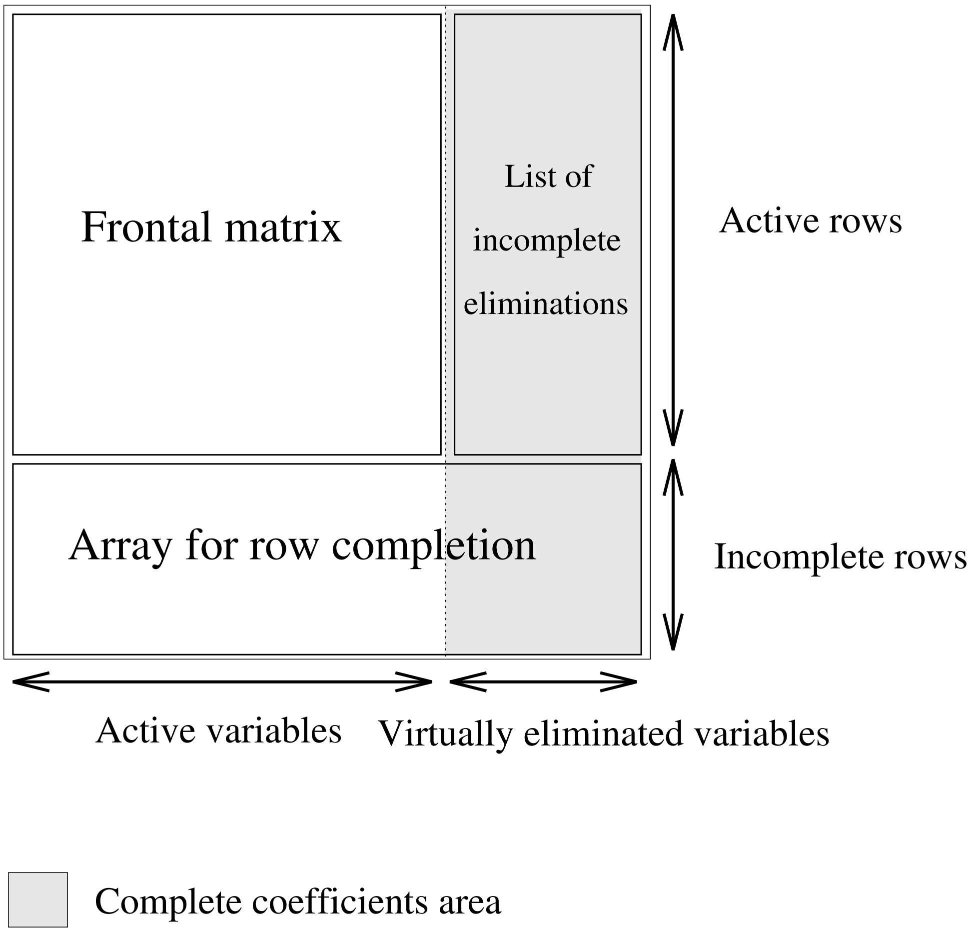


Figure 8: Extended frontal matrix

In the elliptic case, only two categories remain : A (set of active variables) and F (eliminated ones). Introducing this partial pivot, we obtain an $(A+B+C)-by-(A+D)$ frontal matrix, an $(A+B+C)-by-(B+E)$ list of incomplete eliminations and a $D-by-(A+D+B+E)$ array for row completion. So the extended frontal matrix M will be an $(A+B+C+D)-by-(A+D+B+E)$ array.

What kind of control do we have over M's size ? We shall show that, during the computation, the sizes of previous sets verify

$$\#D = \#B + \#C = \#B + \#E$$

So M is a square array, and if n is the number of "elliptic" variables,

$$\#M \leq 4n^2$$

4.3.2 Practical algorithm

More precisely, we shall note $x : A \to B$ the passage of variable x from category A to category B. We will also note $pivot(x)$ the index of the row (and the corresponding variable) chosen to eliminate x fictively.

$$pivot : (B + E) \mapsto D$$

So let us eliminate variable x.

(1) Complete column x (note that we first store results in an array m).

$$\begin{array}{lll} \text{Initialisation} & m_y := M_{y,x} & \forall y \in (A + B + C + D) \\ \text{Completion} & m_y := M_{y,x} - \frac{M_{y,z} \times M_{pivot(z),y}}{M_{pivot(z),z}}, & \left\{ \begin{array}{l} \forall y \in (A + B + C) \\ \forall z \in (B + E) \end{array} \right. \end{array}$$

(2) Do the pivot choice : we note the row $y \in (A + B + C)$.

(3) Proceed to elimination. Two cases are possible.

- $y \in A$ ie y active variable. Then we make virtual elimination; $x : A \to B$ or $x : D \to E$, $y : A \to D$ with $pivot(x) = y$. And we have to update :

$$M_{.,x} := m$$

- $y \in (B + C)$ ie y passive variable. Then elimination can be completed. We have the following transfers among categories :

$$\begin{array}{lcl} x : A \to C & \text{or} & x : D \to F \\ y : B \to E & \text{or} & y : C \to F \end{array}$$

  We have also to proceed to concrete elimination

$$M_{\alpha,\beta} = M_{\alpha,\beta} - \frac{M_{\alpha,x} \times M_{y,\beta}}{M_{y,x}} \quad \left\{ \begin{array}{l} \forall \alpha \in (A + B + C + D) \\ \forall \beta \in (A + D) \end{array} \right.$$

  and there we shall use the fact that $m_z = M_{z,x}$.

(0) Perform (now ready) postponed eliminations

- Add newly assembled coefficients to frontal matrix and array of row completion.
- If $x \in D$, proceed to effective elimination of variable z such that $pivot(z) = x$.

$$M_{\alpha,\beta} = M_{\alpha,\beta} - \frac{M_{\alpha,z} \times M_{x,\beta}}{M_{x,z}} \quad \left\{ \begin{array}{l} \forall \alpha \in (A + B + C) \\ \forall \beta \in (A + D) \end{array} \right.$$

We summarise the impact of these variable movements on $\#D$, $\#B + \#C$, and $\#B + \#E$ in the following table :

Case	Movement	#A	#B	#C	#D	#E	#F	#D	#B+#C	#B+#E
$y \in A$	$x : A \rightarrow B$	-1	+1							
	$y : A \leftarrow D$	-1			+1			+1	+1	+1
	$x : D \leftarrow E$				-1	+1				
	$y : A \leftarrow D$	-1			+1			=	=	=
	$z : E \leftarrow F$					-1	+1			
$y \in B + C$	$x : A \rightarrow C$	-1		+1						
	$y : B \leftarrow E$		-1			+1		=	=	=
	$x : D \leftarrow F$				-1		+1			
	$y : B \leftarrow E$		-1			+1		-1	-1	-1
	$z : E \leftarrow F$					-1	+1			
	$x : A \rightarrow C$	-1		+1						
	$y : C \leftarrow F$			-1			+1	=	=	=
	$x : D \leftarrow F$				-1		+1			
	$y : C \leftarrow F$			-1			+1	-1	-1	-1
	$z : E \leftarrow F$					-1	+1			

This algorithm shows that in fact managing local pivot introduces many backward calculations. However, while computing time at most four times longer, an average behaviour should lead us to a cost of 1.5×1.5=2.25 times the elliptic one, which seems acceptable.

Another interesting comment concerns variable managing. We noticed that embarrassing coefficients of variables were stored in columns B and C of the matrix. We have to note that these rows are either pivot remembering or dependence coefficients. So it is perfectly possible that variable x starts being virtually eliminated, then that y is completely eliminated before x's problem is solved. Indeed, coefficients of the array of row completion have a very different meaning to those of the frontal matrix. Moreover backward substitution will be slightly modified. It will include a progressive "building" of variables during the time separating virtual elimination from effective one ...

Meanwhile, the problem of double elimination is still not solved. The next paragraph gives a means to continue. A more sophisticated method is given afterwards.

4.4 A Simple Mean To Prevent From Double Elimination

To avoid eliminating the same row twice, we just have to put a flag on all these rows, which each of these may acquire if nobody did it before.

So when a processor would like to use a row for elimination, it will have to make a request. If it is not accorded, it will restore pivot column. Then the virtually eliminated variable will stay "active".

Now it is difficult to predict how many additional variables will be hence produced. Many variables may ask for the same locked row. However, let us consider that these choices occur independently. Then we notice that the number of "false active" variables should be equal to the number of normally active variables - if the front is totally locked.

But the front is essentially shared and it is probable that each front will ask for only half of the variables. So, if a front is terminated before another, we can assume that only 25% variables may be added to the current front.

Moreover, even if this should be tested, a 25% increase of effective front length will provoke a great loss of performance in fusion times - which is yet the bottleneck of Deepness Analysis. So we have to introduce alternative methods to avoid this.

4.5 Dual Elimination To Prevent From Double Elimination

One may argue that we could put flags so that one variable would be assigned to a unique front. Not only this method will increase the front's size but it will also introduce global communications. Let us focus on a way to avoid this.

Partial Gauss pivot is warranted to be effective by determinant analysis - which specify that at each stage, a non-zero coefficient will be found in the current column.

We shall here be slightly thinner : one is not obliged to use columns for pivoting. A dual alternative consists in using rows for that purpose. The interest we have there, is that if we choose which row will be used for an elimination, we avoid the possible case in which the front of the other side uses frontal rows.

Of course, inside one front, the way to advance must be uniform : we cannot handle both incomplete rows and incomplete columns. That is why a similar front must have a unique elimination mode.

4.6 Total fusion

These ideas are particularly well-fitted to the bifrontal scheme; each of the two fronts may use a different elimination mode and final elimination (ie front fusion) will be done without any problem of double elimination.

How to extend this result to multifrontal schemes ?

Example (#nodes)	King alg.	Minimum degree	*Deepness Analysis*	Fusion Share	%
1 (295)	67538	15744	20789	6789	32.7%
2 (171)	16393	7729	5190	738	14.2%
3 (1015)	622922	223973	222180	69442	31.3%
4 (400)	82537	53803	42789	4466	10.4%
5 (553)	133160	115374	91289	9717	10.6%
6 (253)	21786	15484	9528	564	5.9%

Figure 9: Deepness Analysis compared to other reordering schemes

4.7 Partial fusion - Bicoloring problem

Partial fusion is more difficult to handle. The problem is that they make a mixed front appear which is then totally available to move. We have to include some post-frontal variables, or even use a third front of appropriate mode to escape from this.

So each spine node should have an even number of parallel fronts. We may then notice that such a way of working will increase the total number of variables to fusion - simply because each 3-front node has to be replaced by a 4-front node. However, elimination is then much easier and some more parallelism is introduced.

Transforming each odd node into an even one, choosing for each front one of the dual elimination modes belongs to a well-known problem : bicoloring.

5 Experimental results

5.1 Sequential tests

It is interesting to see what kind of improvement we get, regardless of the machine (or architecture) used. So here we will focus on the efficiency parameter (see [18, 17]), and compare for some examples our efficiency to the King's algorithm one ([13]). To be more honest, all vertices have been tested as pseudo-peripheral nodes, so that finding a "good" pseudo-diameter should not been taken into account in the use of this scheme. Of course, we give the best performance among all these various choices.

In order to give a better idea on how parallelisable is the new algorithm, we also give which part of the efficiency parameter is concerned with the front fusion process.

In figure 9, matrices 1 and 3 are purely artificial triangulated planar graphs. Matrix 2 comes from a real-life problem raised by Hydro-Quebec : it is a modelisation of the "Saint-Laurent" used, for instance, to study the propagation of pollution. Triangulations 4, 5 and 6 are issued from a Delaunay algorithm and are typically regular.

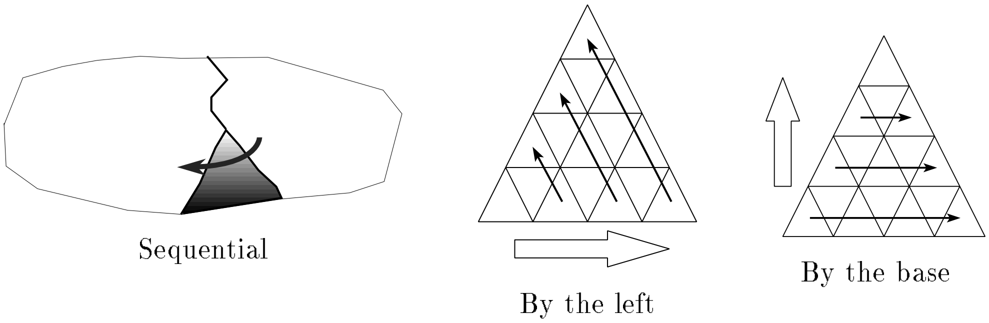


Figure 10: sequential vs parallel scheme

We notice that not only *Deepness Analysis* improves the efficiency parameters but also tends to reduce front fusion for regular cases. The sequential part is not a simple mono-processor phase : the computation is shared by only two processors so that meanwhile some other tasks may be performed. However, the parallelism directly introduced by *Deepness Analysis* leads definitively to coarse-grained parallelism.

In the next section, we shall explain why parallel fronts are improving efficiency parameter. We stress artificial parallel fronts are also a way to improve parallelism and to prevent from double-elimination.

5.2 The impact of parallel fronts

We shall present this theoretical problem as an alternative of "normal" optimal fronts going along a regular area of the mesh. Figure 10 explains the process in a more visual way. For this, we tried two main approaches on the regular parallel area. In fact, the first mode is describing an "optimal-fronts-like" method, while the second one is more original. Of course, we must not forget that it is possible to apply George's results on this scheme, and we shall get some better results in assymptotical terms.

We shall consider the efficiency parameter for variables linked to vertices. So we obtain the formula and assymptotical behaviours shown in figure 11. Of course, nested dissection will lead to a $O(q^3)$ behaviour, but requires at least a similar mesh in terms of regularity. However, we note that these methods lead to coarse-grained parallelism.

6 Conclusion

Today, it is difficult to see how helpful *Deepness Analysis* is for common-life meshes. Even if theoretically, it costs $O(n^2)$, we saw that its latency constant is completely comparable with

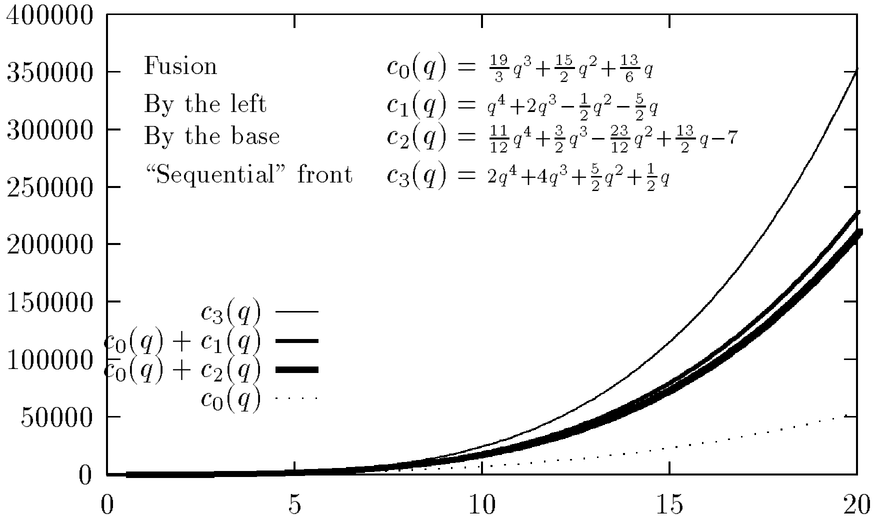


Figure 11: Assymptotical behaviour

normal frontal methods and using George's considerations on regular meshes, we can draw an interesting scheme for each triangulation with $O(n^{1.5})$ convergence rate.

As we said at the beginning, using triangles is probably more an advantage than an inconvenience of our method. However, it could be interesting to see if such schemes have an interest for other kinds of domain structures like quadrangulations. Of course, we could transform the quadrangulation into a triangulation and introduce some additional degrees of liberty to give new elements a "variable-independent" form.

So the main challenge is to export these ideas to the third dimension. In particular, the main notions on connectivity remain but many hard problems may appear; if a very-strong component of the colour n is simply connected, that does not imply that its generated very-strong components of the colour $n+1$ will be simply connected, and we know the importance of simple-connectivity in the demonstrations given here.

Acknowledgements

I would like to thank Laurie Hendren for her most helpful comments. I am also grateful towards the ACAPS lab., and more particularly towards P.Panangaden, C.Verbrunge, and R.Duquette. The careful reading of Jacko Koster and the referee definitely improved the presentation of this document.

References

[1] J.J. Dongarra, I.S. Duff, D.C. Sorensen and H.A. Van der Vorst. Solving Linear Systems on Vector and Shared Memory Computers, 1991.

[2] I.S. Duff. Parallel implementation of multifrontal schemes. *Parallel Computing*, 3:193-204, 1986.

[3] I.S. Duff, A.S. Erisman and J.K. Reid. Direct Methods for Sparse Matrices. *Oxford Science Publications*, 1986.

[4] I.S. Duff and J.K. Reid. The multifrontal solution of indefinite sparse symmetric linear systems. *ACM Trans. Math. Softw.*, 9:302-325, 1983.

[5] I.S. Duff and J.K. Reid. The multifrontal solution of unsymmetric sets of linear equations. *SIAM J. Sci. Stat. Comput.*, 5:633-641, 1984.

[6] I.S. Duff and J.A. Scott. MA42 - A new frontal code for solving sparse unsymmetric systems. *Rutherford Appleton Laboratory*, Report RAL-93-064, September 1993.

[7] D.J. Evans. The use of preconditioning in iterative methods for solving linear equations with symmetric positive definite matrices. *J. Inst. Maths. Applics.*, 4:295-314, 1967.

[8] A. George. Nested dissection of a regular finite element mesh. *SIAM J. Numer. Anal. Vol.10 No2*, April 1973

[9] A.J. Hoffman, M.S. Martin and D.J. Rose. Complexity bounds for regular finite difference and finite element grids. *SIAM J. Numer. Anal.*, Vol.10 No2, April 1973.

[10] P. Hood. Frontal Solution Program for Unsymmetric Matrices. *Int. J. num. Meth. Engng.*, 10:379-399, 1976.

[11] B.M. Irons A Frontal Solution Program for Finite Element Analysis. *Int. J. num. Meth. Engng.*, 2:5-32, 1970.

[12] D.A.H. Jacobs. Preconditioned conjugate gradient methods for solving systems of algebraic equations. Note No. RD/L/Nl93/80, CERL, Leatherhead, 1980.

[13] P. King. Automatic reordering scheme for simultaneous equations derived from network systems. *Int. J. num. Meth. Engng.*, 2:523-533, 1970.

[14] R.J. Lipton, D.J. Rose and R.E. Tarjan. Generalized nested dissection. *SIAM J. Numer. Anal.*, Vol.16, No 2:346-358, 1979.

[15] J.K. Reid. Frontal methods for solving finite-element systems of linear equations. *Sparse Matrices and Their Uses*, I.S. Duff(editor), Academic Press Inc. (London) LTD, 1981.

[16] D.J. Rose. Triangulated graphs and the elimination process. *J. of Math. Anal. and Appl.*, 32:597-609, 1970.

[17] S.W. Sloan and M.F. Randolph. Automatic Element Reordering for Finite Element Analysis with Frontal Solution Schemes. *Int. J. num. Meth. Engng.*, 19:1153-1181, 1983.

[18] W.F. Tinney and J.W. Walker. Direct solutions of sparse network equations by optimally ordered triangular factorization. Proc. of the IEEE, Vol.55, No11, November 1967.

[19] M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM J. Alg. Disc. Meth.*, Vol.2, No1, March 1981.