Titre: Title:	Résolution d'un problème aux limites à frontière libre au moyen d'un algorithme de remaillage adaptatif et anisotrope
Auteur: Author:	Éric Béchet
Date:	2002
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Béchet, É. (2002). Résolution d'un problème aux limites à frontière libre au moyen d'un algorithme de remaillage adaptatif et anisotrope [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7084/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7084/
Directeurs de recherche: Advisors:	François Trochu, & Jean-Christophe Cuillière
Programme: Program:	Non spécifié

UNIVERSITÉ DE MONTRÉAL

RÉSOLUTION D'UN PROBLÈME AUX LIMITES À FRONTIÈRE LIBRE AU MOYEN D'UN ALGORITHME DE REMAILLAGE ADAPTATIF ET ANISOTROPE

ÉRIC BÉCHET
DÉPARTEMENT DE GÉNIE MÉCANIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIAE DOCTOR
(GÉNIE MÉCANIQUE)
NOVEMBRE 2002

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

RÉSOLUTION D'UN PROBLÈME AUX LIMITES À FRONTIÈRE LIBRE AU MOYEN D'UN ALGORITHME DE REMAILLAGE ADAPTATIF ET ANISOTROPE

présentée par: BÉCHET Éric

en vue de l'obtention du diplôme de : Philosophiae Doctor

a été dûment accepté par le jury d'examen constitué de:

M. CAMARERO Ricardo, Ph.D, président

M. TROCHU François, Ph.D, membre et directeur de recherche

M. CUILLIÈRE Jean-Christophe, Ph.D, membre et codirecteur de recherche

M. LE ROUX Daniel, Ph.D, membre

M. GUIBAULT François, Ph.D, membre

DÉDICACE

À mes parents, ma famille.

REMERCIEMENTS

En premier lieu, je tiens à remercier mon directeur de recherche, M. François Trochu, ainsi que mon codirecteur, M. Jean-Christophe Cuillière, pour leur soutien constant lors de la réalisation de cette thèse.

Cette thèse a été l'occasion pour moi de passer quelques années au sein du centre de recherches appliquées sur les polymères de l'École Polytechnique de Montréal où j'ai côtoyé de nombreuses personnes, qu'elles soient remerciées ici pour leurs conseils, leur amitié,... ; en particulier et sans ordre particulier, Tan, Élaine, Jean-François, Ludovic, Eduardo, Christian, Bart.

Mes remerciements vont également aux organismes FCAR et CRSNG pour l'aide financière apportée.

Enfin, je souhaite remercier Messieurs Ricardo Camarero, Daniel Le Roux et François Guibault pour avoir accepté de juger cette thèse.

RÉSUMÉ

Cette thèse est consacrée à l'étude et au développement d'algorithmes de remaillage dans le cadre de la simulation d'écoulements à surface libre en milieu poreux. Ces écoulements se rencontrent en particulier dans la phase de remplissage d'un moule pour pièces en matériaux composites par le procédé RTM (Resin Transfer Moulding ou moulage par transfert de résine), pour lequel la majorité des applications de cette recherche ont été faites. Les méthodes de remaillage présentées ici sont basées sur le critère de Delaunay pour les triangulations. Une adaptation aux surfaces courbes discrètes est proposée. Cette adaptation permet de s'affranchir de la représentation CAO exacte des surfaces et de n'utiliser qu'une représentation approchée par une triangulation sommaire. En effet, la possibilité de générer des maillages anisotropes surfaciques sans garder le lien avec l'outil ayant servi pour modéliser ces surfaces permet un interfaçage aisé avec les solveurs par éléments finis. Cette partie a donné lieu à une première publication. Pour les problèmes pratiques rencontrés en RTM, les idées de cette première partie on été réutilisées et modifiées pour permettre de générer un maillage anisotrope. Le critère de Delaunay considéré ici est donc anisotrope. En effet, la méthode de remaillage présentée dans cette thèse permet de mieux simuler l'évolution du front de matière en permettant l'aplatissement des éléments dans le sens de propagation de l'écoulement. Ainsi, la représentation de la frontière de l'écoulement est très régulière et lisse, ce qui n'était pas le cas des simulations faites sur un maillage fixe et isotrope à nombre de degrés de liberté égal. Pour l'évolution de la surface libre dans le temps, la méthode de remaillage a été couplée originalement avec une approche par *level-sets* de façon à contrôler le pas de temps indépendamment du maillage utilisé. Cette partie a donné lieu à une seconde publication. L'adaptation aux simulations thermiques est proposée dans la troisième partie de cette thèse. En effet, le remaillage permet aussi de mieux contrôler la diffusion numérique (artificielle) induite par les méthodes de résolution en représentation Eulerienne utilisées lorsque des phénomènes

de transport sont en jeu. Cette diffusion est liée au pas de discrétisation spatial. Dans le cas du RTM, il arrive fréquemment que, au voisinage du front de résine et quand la température diffère de celle du moule et des fibres, la diffusion numérique soit un obstacle au calcul précis des condition thermiques dans le moule. Les équations de transport requièrent aussi la satisfaction d'une condition sur le pas de temps afin de rendre le schéma numérique stable dans le cas d'une résolution sur grille Eulerienne. Une approche à pas de temps variable pour les seuls calculs de transport est proposée. Une étude visant à prouver la possibilité de générer un maillage unique pour l'ensemble de la simulation est tentée pour les géométries bidimensionnelles (planes). Un estimateur d'erreur basé sur la matrice Hessienne en pression est utilisé pour permettre la génération d'un maillage rendant l'erreur d'interpolation uniforme. En parallèle, un heuristique est conçu pour étirer les éléments en fonction des positions successives du front déterminées *a priori*, afin de permettre une meilleure approximation du front lors de la simulation. Enfin, une étude analytique d'un cas d'injection montre les difficultés à générer un maillage satisfaisant deux conditions : uniformité de l'erreur d'interpolation et uniformité du nombre de Courant (minimisation de la diffusion dans les équations de transport et condition de stabilité). Ceci a donné lieu à la publication d'un troisième article. La dernière partie de cette thèse est l'application aux surfaces courbes de l'estimateur d'erreur. Les surfaces considérées ici sont discrètes et il est en effet nécessaire de discriminer l'erreur d'interpolation de l'erreur géométrique, afin d'éviter de raffiner inutilement le maillage à chaque « cassure » de la géométrie. Une application sur une géométrie issue de l'industrie est présentée. Cette partie a donné lieu à un quatrième article. Enfin, une discussion sur l'ensemble de la recherche menée dans cette thèse est présentée.

ABSTRACT

This thesis focuses on the study and development of remeshing algorithms for the simulation of free surface flows in porous medium. This kind of flow is coming up with the mould filling phase when manufacturing composite parts with the RTM process (Resin Transfer Moulding), for which the majority of the applications of this research have been done. Remeshing methods presented here are based on the Delaunay criterion for triangulations. An adaptation for curved surfaces is proposed here. This adaptation avoid to keep the link with an exact representation of the CAD surface, and allows the use of a simple tessellation, instead. In fact, the ability to generate anisotropic surface meshes without keeping the link with the tool used to model those surfaces allows to interface finite element solvers with ease. A first publication has been made, based on this part. For practical problems arising in RTM, the ideas coming from the first part have been adapted to the goal of generating anisotropic elements. Thus, the Delaunay criterion considered here is anisoptropic. In fact, the remeshing method presented in this thesis allows a better simulation of the advancing flow front by using anisotropic elements flattened in the direction of the flow. Thus, the resolution of the flow front is high an shows a smooth and regular front. This was not the case with simulations made on a fixed and isotropic mesh, with the same number of degree of freedom. For the evolution of the free surface in time, a level-set approach was originally combined with the remeshing algorithm in order to control the time step independently to the mesh. A second publication has been based on this part. An adaptation of the remeshing algorithm is proposed for thermal problems in the third part of this thesis. In fact, the remeshing allows a better control of the numerical (artificial) diffusion that arise while solving a transport phenomenon. This diffusion is related to the spatial discretisation step. In the case of RTM it happens frequently, in the vicinity of the flow front and when temperature differ notably to the temperature of the mould and the fibres, that the numerical diffusion is an obstacle preventing to achieve a precise simulation of the

thermal behaviour in the mould. Transport equations require also a condition on the time step to stabilize the numerical scheme when solved on an Eulerian grid. A variable time-stepping for the sole calculation of transport phenomenon is proposed. A study aimed to prove the possibility of generating a fixed mesh for the whole simulation is proposed for planar geometries. An error estimator based on the Hessian matrix in pressure is used to generate a mesh that will make the interpolation error uniform in the domain. At the same time, an heuristic is built to stretch elements in function of the successive positions of the mesh, determined *a priori*. This is improving the approximation of the front that is made during the simulation. Finally, an analytical study of a injection case is done, showing the difficulty to generate a mesh satisfying two conditions : uniformity of the interpolation error, and uniformity of the Courant number (which is shown to minimize the numerical diffusion in transport equations). This work was published in a third article. The last part of this thesis focuses on the application of the error estimator to curved surfaces. The surfaces considered here are discrete, it is necessary to separate the interpolation error from the geometrical error. This is done to avoid useless refinement of the mesh near angles in the geometry. An sample case from industry is studied. This part composes the fourth article. Finally, a discussion on the whole research made in this thesis is presented.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES ANNEXES

INTRODUCTION

Les méthodes numériques sont aujourd'hui appliquées dans les domaines les plus variés, en raison de leur grande souplesse. Les progrès constants de l'informatique et des ordinateurs permettent de traiter des problèmes de plus en plus complexes. Il est courant de nos jours de traiter des problèmes de simulation non stationnaires, c'est-à-dire dans lesquels l'aspect temporel intervient. Dépendamment de leur taille, ces derniers peuvent nécessiter de grosses capacités et de longs temps de calcul. De tels problèmes à plusieurs centaines de millions de degrés de liberté sont, une fois discrétisés, calculables en quelques semaines sur une machine parallèle. Un des aspects qui influent beaucoup la qualité des solutions obtenues est la discrétisation du problème. Une discrétisation très fine et uniforme dans l'espace et le temps permet généralement d'obtenir une solution de bonne qualité, en autant que la méthode de calcul converge, mais les temps de calcul seront très longs. Il est alors bien souvent nécessaire d'effectuer une discrétisation judicieuse du domaine, dans laquelle seules les zones à fortes variations sont finement représentées, les autres l'étant de façon beaucoup plus grossière. En améliorant ainsi le taux de convergence (rapport entre l'erreur et la taille du problème discret), on gagne plusieurs ordres de grandeur sur le temps de calcul et par conséquent, sur la qualité de la solution à temps de calcul constant.

La façon de discrétiser un problème dépend de la méthode numérique utilisée. La méthode des différences finies par exemple ne suppose qu'une discrétisation spatiale (maillage) et non fonctionnelle puisque la notion d'interpolation ne fait pas partie de la formulation. De plus, elle implique que le maillage soit structuré et dispose d'une connectivité (voir la Figure 0.1 partie supérieure). Cette dernière propriété la rend très peu souple. D'autres méthodes admettent une discrétisation non seulement géométrique mais aussi fonctionnelle, comme la méthode des intégrales de frontières ou la méthode des éléments finis (MEF) qui est au centre de ce travail. Dans cette dernière, le maillage

peut très bien ne pas disposer de connectivité et de notion de voisinage implicites, on parle alors de maillage non structuré (voir la Figure 0.1 partie inférieure). C'est une caractéristique très appréciable lorsque l'on doit traiter des problèmes de géométrie complexe, qui sont fréquemment rencontrés dans l'industrie.

Ce maillage sert de support à l'espace fonctionnel dans lequel la solution du problème est recherchée. Chaque élément du maillage (triangle, quadrilatère etc...) sert de support à l'interpolation et on associe un ensemble de fonctions de forme qui vont permettre d'approcher la solution exacte du problème dans cet élément. Le maillage est donc une composante à part entière de la méthode numérique. et le respect de certaines règles lors de sa construction permet de minimiser les erreurs lors du calcul.

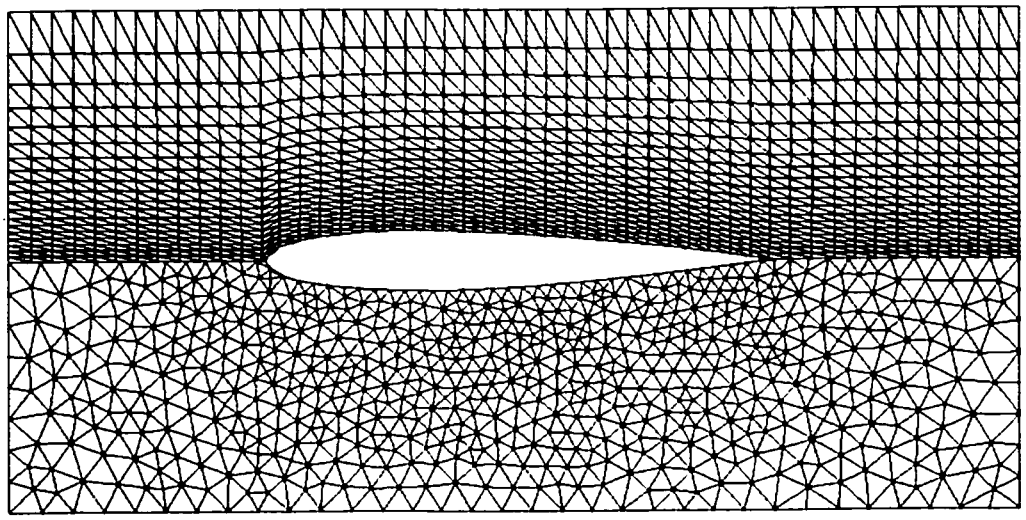


Figure 0.1: Maillage structuré sur la moitié supérieure du domaine: la connectivité est implicite. En bas : maillage non structuré, dans lequel la connectivité est explicite

Il existe de nombreuses variantes de maillages selon la nature du domaine étudié et sa dimension. Pour des problèmes unidimensionnels un maillage est constitué par une succession triviale de segments reliant des sommets consécutifs entre eux. En deux dimensions, on peut trouver des maillages formés d'éléments triangulaires, d'éléments à

quatre côtés (quadrilatères) ou d'une combinaison des deux. En trois dimensions les éléments peuvent être des tétraèdres, des pyramides, des prismes, des hexaèdres. Actuellement les maillages les plus répandus sont constitués de simplexes, i.e. d'éléments les plus simples et de volume (surface en deux dimensions) non nul. En fonction de la dimension, il s'agit de triangles ou de tétraèdres. Il est en effet beaucoup plus facile de paver un domaine à l'aide de simplexes, plutôt qu'à l'aide d'éléments non simpliciaux (quadrilatères ou autres).

Dans la MEF, il est possible d'augmenter ou de mieux répartir le nombre de degrés de liberté d'une simulation dans l'objectif de réduire l'erreur d'approximation. Ceci est réalisé en enrichissant la base fonctionnelle par des fonctions polynomiales d'ordre p supérieur. Celle ci sert à mieux approcher la solution dans chaque élément géométrique et on parle alors de raffinement "p". On peut aussi augmenter le nombre d'éléments dans le maillage, en diminuant leur taille h. On parle alors de raffinement "h". Il reste une alternative : déformer le maillage de façon à mieux répartir les degrés de liberté, sans en augmenter le nombre total : c'est le raffinement "r". Le raffinement "p" permet d'améliorer la précision des simulations dans le cas de phénomènes de nature continue, mais est totalement impuissant face à des phénomènes aux variations brutales tels les chocs aérodynamiques ou le suivi de front. Ceci est du à la nature continue de la base fonctionnelle; au sein d'un élément on ne peut en effet pas rendre compte correctement d'une discontinuité (phénomène de Gibbs lié à la projection L^2, voir la Figure 0.2). Il est de plus très difficile d'orienter la base fonctionnelle de façon à "capter" des effets directionnels. Les raffinements "h" et "r" sont par contre très adaptés à ce style de simulation, et ils permettent de plus d'imposer une anisotropie dans le maillage (voir la Figure 0.4). C'est la solution généralement retenue pour traiter les chocs et discontinuités.

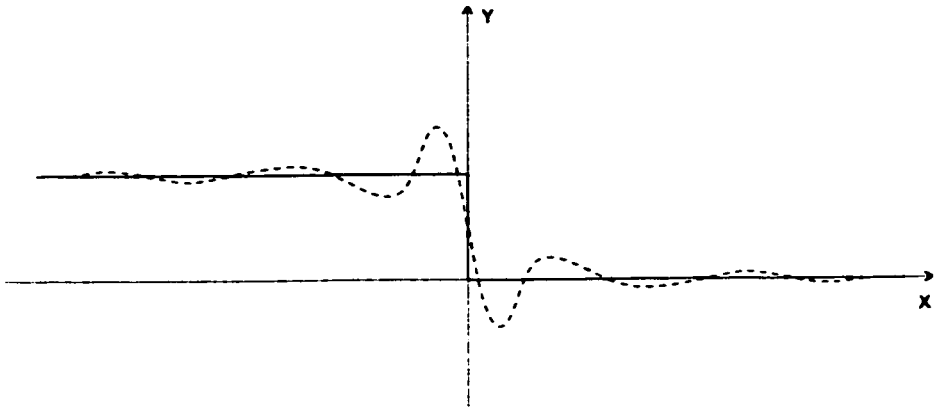


Figure 0.2: Phénomène de Gibbs unidimensionnel. Cet effet se produit lorsque l'on essaie de faire la projection d'une fonction discontinue (ici la marche en trait plein) dans un espace fonctionnel polynomial. Le projeté est la courbe oscillante

Dans le cadre de phénomènes non stationnaires, ces chocs et ces discontinuités et d'une façon générale les caractéristiques de la solution, vont se déplacer avec le temps. Il est donc nécessaire de modifier la discrétisation en cours de calcul pour tenir compte de l'évolution du phénomène. Cette adaptation de la discrétisation permet d'éviter des durées de calcul prohibitives (ou des résultats peu précis) liés à l'utilisation d'une discrétisation fixe, et d'obtenir de bons taux de convergence. Le développement de méthodes de maillage et de remaillage adaptées aux phénomènes non stationnaires et en particulier aux phénomènes impliquant un front mobile, par exemple la frontière entre deux fluides non miscibles, constitue l'un des objectifs de cette thèse.

Figure 0.3: Front de matière en mouvement (en deux dimensions)

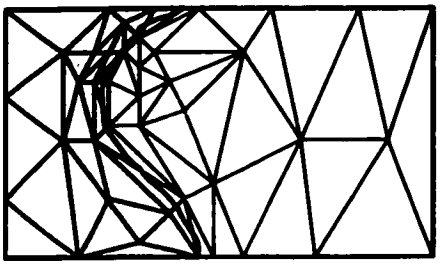


Figure 0.4: Maillage non structuré qui "suit" le front de matière (utilisation d'éléments anisotropes et adaptés en taille)

Le Chapitre 1 est une étude bibliographique mettant en valeur les travaux précédents dans le domaine de la génération et l'adaptation de maillages. L'objectif de cette étude est de situer le travail dans le cadre des développements les plus récents, notamment sur les aspects anisotropes et les frontières mobiles. Le Chapitre 2 présente l'organisation générale de la thèse par articles. Le Chapitre 3 présente un algorithme de génération de maillage pour surfaces courbes et discrètes. Le Chapitre 4 présente un algorithme de génération de maillages adaptés aux écoulement à frontière libre rencontrés dans le procédé RTM. L'évolution du front est gouvernée par un algorithme basé sur les level-sets originalement associé à l'algorithme de remaillage. Le Chapitre 5 présente une extension dans le cas d'un problème de transport (simulations thermiques), et un algorithme permettant de générer un maillage unique pour l'ensemble de la simulation à l'aide d'un estimateur d'erreur. Le Chapitre 6 présente une extension aux surfaces courbes de l'estimateur d'erreur du Chapitre 5, et une application pour une pièce réelle provenant de l'industrie. Enfin, une discussion générale sur l'ensemble le la recherche présentée dans ce mémoire clôt la thèse.

CHAPITRE 1
REVUE BIBLIOGRAPHIQUE

1.1 Introduction

Les techniques de maillage et de remaillage sont très diverses. Historiquement, les premières à faire leur apparition sont les méthodes fondées sur la décomposition manuelle du domaine en zones plus simples à mailler, le maillage de chacune de ces zones (souvent des quadrilatères en 2D et des hexaèdres en 3D) étant effectué automatiquement selon un schéma prédéfini. On ne pouvait alors pas parler de maillage automatique, puisque l'essentiel du travail était effectué manuellement.

Les méthodes utilisées pour générer des maillages structurés (maillages obtenus par résolution d'équations aux dérivées partielles principalement) ne permettent pas d'atteindre une souplesse suffisante pour les problèmes que nous rencontrerons. Nous ne nous attarderons donc pas plus sur ces méthodes (Camarero et Reggio, 1983; Thomson et al, 1985; Winslow, 1967).

Aujourd'hui, il n'existe que quelques grands types de mailleurs automatiques : ceux utilisant la méthode dite de Delaunay (Borouchaki et al., 1997; Cigoni et al, 1998; Delaunay, 1934; George et Borouchaki, 1997) (probablement la plus répandue), ceux fondés sur l'avance de front (Cuillière, 1998; François, 1998; Löhner et Parikh, 1988), les méthodes opérant par décomposition spatiale (énumération exhaustive (Thacker et al., 1980), quadtree et octree (Yerry et Shephard, 1984)), et enfin les méthodes basées sur la compaction de sphères (Shimada, 1993) ("sphere packing" ou encore "bubble packing"). Chacune de ces méthodes se distingue par un grand nombre de variantes. Dans ce qui suit, nous ne nous intéressons qu'aux mailleurs entièrement automatiques et

aux méthodes de remaillage existantes. Dans chaque cas, nous étudierons les développements les plus récents concernant l'aspect anisotrope et le suivi d'un front.

1.2 Méthodes de maillage existantes

1.2.1 Mailleurs de Delaunay

Les mailleurs dits "de Delaunay" permettent de trianguler (dans le sens de paver à l'aide de simplexes) un domaine isomorphe à $\Re^n$. A priori le domaine doit être convexe, mais c'est une condition extrêmement contraignante. Les éléments du maillage sont des simplexes (triangles en deux dimensions, tétraèdres en trois dimensions, etc). Il est bien entendu possible de généraliser ces méthodes de maillage aux domaines non convexes et en respectant des contraintes (arêtes et faces devant faire partie de la triangulation). La façon de générer les simplexes est fondée sur les résultats de Dirichlet (1850), Voronoï (1908) et bien entendu Delaunay (1934). Dirichlet montre que l'on peut, en deux dimensions, faire une partition du plan en cellules convexes (et polygonales) à partir d'un nuage de points en se basant sur des critères de proximité. Voronoï étend cette partition en dimension quelconque, on parle alors de partition en cellules (ou *d*-polytopes, *d* étant la dimension de l'espace) de Voronoï. La triangulation de Delaunay d'un nuage de points est en fait le dual de cette partition en cellules convexes. L'idée qui sous-tend tous les mailleurs dits "de Delaunay" est qu'il faut construire le maillage en respectant à chaque étape le critère de Delaunay appelé "*critère de la sphère vide*". Un maillage qui est "*de Delaunay*" possède des propriétés intéressantes comme nous le verrons par la suite, en particulier en deux dimensions.

1.2.1.1 Théorie sous jacente

Soit un ensemble de points $V = \{v_1,...,v_n\}, n \geq d+1$ dans l'espace euclidien E^d (cet espace est de dimension *d*). Si $dist(v_i, v_j)$ désigne la distance euclidienne entre les points v_i et v_j, la région $V(i) = \{x \in E^d | dist(v_i, x) \leq dist(v_i, v_j), j = 1...n\}$ qui est le lieu des points plus proches de v_i que tout autre point de V est appelée le *d-polytope de Voronoï* (ou *cellule de Voronoï*) associé au point v_i. Si *d*=2 nous sommes ramenés dans le cas bidimensionnel, plus commode pour les schémas (voir la Figure 1.1). Si on relie le nœud interne à chaque cellule de Voronoï aux nœuds des cellules immédiatement voisines, on obtient le maillage appelé *triangulation de Delaunay* (voir la Figure 1.2). Cette construction est unique si les points sont en configuration générale, i.e., l'ensemble V n'admet pas *d*+2 points cocycliques. Dans le cas contraire il existe plusieurs maillages équivalents qui seront abusivement appelés "*triangulations de Delaunay*".

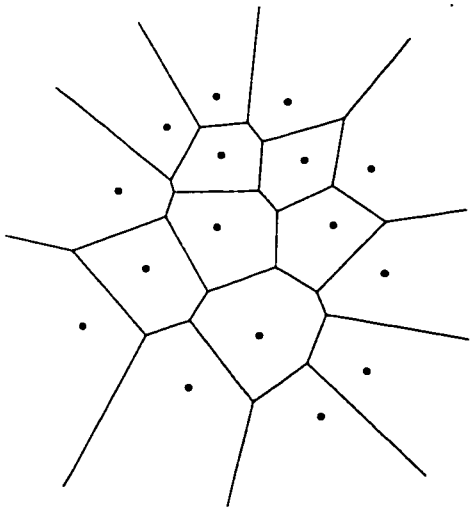


Figure 1.1: Diagramme de Voronoi en deux dimensions

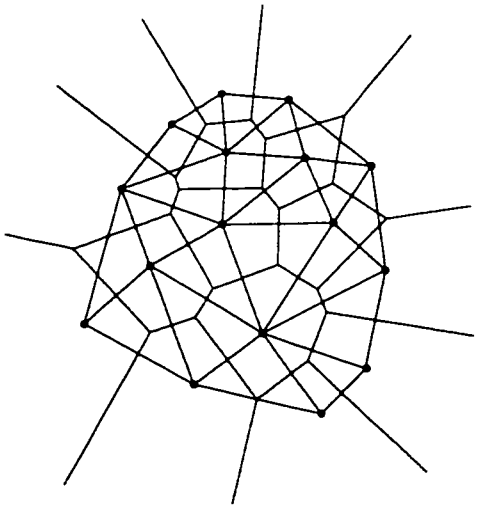


Figure 1.2: Diagramme de Voronoi et triangulation de Delaunay

Le critère de Delaunay, ou critère de la sphère vide, dit que pour un maillage de Delaunay, la sphère ouverte circonscrite à chaque simplexe ne contient strictement aucun point de V. Comme nous le verrons par la suite, ce critère est intimement lié à la notion de distance dans l'espace considéré. Pour le moment, nous en restons à la notion de distance euclidienne.

Le lemme général de Delaunay démontré en 1934 (Delaunay, 1934) est à l'origine des algorithmes de génération de maillages de Delaunay.

Lemme général de Delaunay. Si T est une triangulation quelconque de l'enveloppe convexe d'un nuage de points V, alors si la propriété de la sphère vide est vérifiée pour toute configuration de deux simplexes adjacents de T, elle est vérifiée globalement et T est une triangulation de Delaunay.

A partir de ce lemme il est facile de montrer que, en deux dimensions, on peut passer d'une triangulation d'un ensemble convexe de points à une autre par retournement d'arêtes (George et Borouchaki, 1997; Cherfils et Hermeline 1990). Ceci n'est toutefois pas établi en trois dimensions (par exemple, une modification de la topologie sans ajout ni retranchement de sommets permet de modifier le nombre de tétraèdres du maillage), ni *a fortiori* en d dimensions. Il est à noter que le retournement d'arête est le seul opérateur topologique en deux dimension (autrement dit, tous les opérateurs se ramènent à une série de retournements d'arêtes). Pour des maillages en dimension supérieure, il existe plusieurs opérateurs topologiques (en trois dimensions, retournements d'arête et de face par exemple) qui rendent délicates les manipulations de ce type de maillage.

Dans le plan, la triangulation de Delaunay maximise l'angle minimal formé par les arêtes des triangles, ce qui la rend très apte à être utilisée pour un calcul par éléments finis. En dimension supérieure, un problème survient, car il est possible d'avoir des éléments de

volume nul qui respectent toutefois le critère de Delaunay (tétraèdres plats appelés "slivers" (Cavendish et al., 1985). Ces éléments sont tout à fait incompatibles avec un calcul par éléments finis. Ils doivent être éliminés après la génération du maillage par une phase d'optimisation.

1.2.1.2 Principe de génération

La plus commune, appelée méthode incrémentale, consiste à insérer les points un à un dans la triangulation, en partant de la triangulation triviale (ne dépendant que de la dimension) d'une boîte englobant tout le domaine considéré. À chaque étape, on cherche à déterminer la triangulation $DT(V_{n+1})$ contenant $n+1$ points à partir de la triangulation $DT(V_n)$ des n points obtenus à l'étape précédente. Le problème élémentaire consiste à insérer un point dans une triangulation qui respecte le critère de Delaunay, de façon à ce que la triangulation issue de cette opération le respecte aussi. Ceci constitue le noyau de Delaunay. L'algorithme dit "de Watson" (Watson, 1981) permet d'effectuer très efficacement cette opération. Cet algorithme consiste en la destruction des simplexes qui ne respectent pas le critère de Delaunay vis à vis du point P à insérer. Ces simplexes sont déterminées à partir de la mesure de Delaunay définie de la façon suivante :

Soit r_K le rayon du cercle circonscrit au simplexe K, soit $dist(P,O_K)$ la distance entre P et le centre O_K de ce cercle. Alors $\alpha(P,K)$ défini par :

$$\alpha(P,K) = \frac{dist(P,O_K)}{r_K} \qquad (1)$$

est la mesure de Delaunay et la condition d'appartenance du simplexe K à la cavité C_P associée à P est :

$$\alpha(P,K) < 1 \tag{2}$$

Tous ces simplexes forment une cavité connexe (Watson, 1981). La cavité C_P doit être vidée de tous ses éléments, il suffit alors d'étoiler le point P avec les frontières de C_P pour obtenir la triangulation de B_P, que l'on doit alors réinsérer dans la triangulation. Formellement, ces étapes se résument en :

$$DT(V_{n+1}) = DT(V_n) - C_P + B_P \tag{3}$$

Ces étapes sont représentées dans les figures ci-dessous.

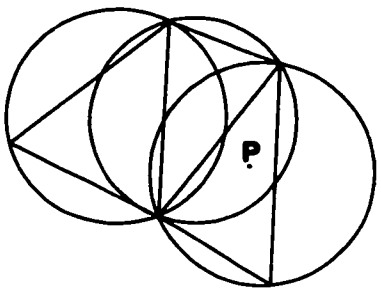


Figure 1.3: Cercles circonscrits et le point P à insérer dans la triangulation $DT(V_n)$

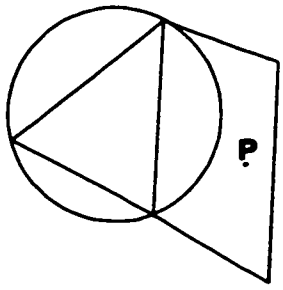


Figure 1.4: Simplexes de C_P détruits

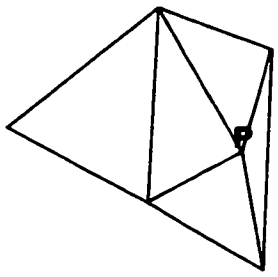


Figure 1.5: Simplexes de B_P construits et ajoutés à la triangulation pour former $DT(V_{n+1})$

On applique ce schéma pour tous les points à insérer pour obtenir finalement la triangulation de Delaunay non contrainte de l'ensemble des points, en plus des points de la boîte englobante. Il suffit de retirer les éléments contenant au moins un point appartenant à cette boîte pour obtenir le maillage de l'enveloppe convexe de l'ensemble des points de V. Il existe plusieurs autres méthodes de génération de maillages dites "de Delaunay". L'approche consistant à "diviser pour régner" (Cigoni et al., 1998; George et Borouchaki, 1997) permet la génération de triangulations de Delaunay quand les points à

insérer sont connus à priori. Cette méthode est plus rapide mais du même ordre en $nLog(n)$ que la méthode précédente.

Les domaines que l'on veut mailler ne sont pas nécessairement convexes. De plus, il est parfois intéressant d'imposer des contraintes sur le maillage, i.e.. d'imposer que le maillage passe par certains endroits (en 2D il peut s'agir d'arêtes et en 3D d'arêtes ou de faces). La frontière non convexe et les contraintes n'ont aucune raison de faire partie du maillage obtenu à l'aide des procédures décrites plus haut, tout simplement parce qu'à aucun moment dans cet algorithme on ne tient compte des relations entre les points que l'on insère (relations d'appartenance à une arrête par exemple). Il existe plusieurs façons de le faire. La première méthode, dite de "forçage des contraintes", consiste à faire apparaître les contraintes dans le maillage uniquement par des opérations topologiques (par exemple retournement d'arêtes en deux dimensions). Poussée à l'extrême, sans même parler d'insertion de points selon la méthode de Delaunay, on retrouve la méthode de Coupez (1991). Cette récupération des frontières fonctionne bien en deux dimensions. En trois dimensions les choses se compliquent car il n'est pas prouvé que l'on puisse passer d'une triangulation à une autre en faisant des inversions de faces ou d'arêtes (Georges et Borouchaki, 1997). Il existe d'autres méthodes qui consistent en un cassage des contraintes, qui a l'avantage d'être valable quelque soit la dimension de l'espace. L'idée est de découper les contraintes qui ne font pas partie de la triangulation finale en chaque point où elles croisent une arrête du maillage. Ceci assure qu'elles feront partie, au moins sous une forme "découpée", de la triangulation finale. Ce cassage est effectué en générant des points à l'intersection de toutes les entités qui coupent la contrainte, et en insérant ces points à l'aide de la méthode classique de Delaunay. Il existe une autre méthode (proposée dans (Pébay et Frey, 1998)) qui consiste à rendre les contraintes et la frontière Delaunay-admissibles *a priori*, c'est à dire avant de commencer à mailler, toutefois elle n'est pour l'instant pleinement valable qu'en deux dimensions. En conclusion, un mailleur purement "Delaunay" en 3D ne respecte pas forcément les frontières de l'objet à mailler, ce qui peut poser quelques problèmes.

1.2.2 Mailleurs frontaux

Les mailleurs frontaux n'existent que pour des espaces isomorphes à $\Re^n$ avec n = 2 ou 3. Il n'existe pas de théorie unifiée concernant ce type de mailleur, car la méthode est fortement empirique et ne repose que sur des notions très intuitives. En deux dimensions, l'application de cette méthode est simple, et elle est généralement reconnue comme fiable. En trois dimensions, peu de recherches ont abouti car il y a beaucoup de problèmes de convergence des algorithmes, en partie à cause du grand nombre de paramètres dont ils dépendent. On peut toutefois citer les travaux de Löhner et Parikh (1988) et de Peraire et al. (1988) qui, bien qu'incomplets ont le mérite d'être les premiers dans ce domaine. Par la suite (et par ordre chronologique), (Golgolab, 1989 ; Lo, 1991 ; Rassineux, 1997) ont développé et stabilisé les algorithmes reposant sur cette approche. Il existe assez paradoxalement une grande variété de principes utilisés lors de l'avance d'un front. Plus curieusement encore, ceux-ci sont parfois totalement antagonistes. Il s'agit là d'une des conséquences de l'aspect empirique de la méthode. Une des aspects intéressants de cette approche est qu'elle conserve naturellement les frontières, contrairement à la méthode de Delaunay dans laquelle il faut effectuer un traitement particulier. Toutefois, cette méthode est une des plus lentes, ceci est dû aux nombreux tests d'intersections nécessaires pour tenter d'assurer un maillage valide (cela ne marche malheureusement pas systématiquement !)

1.2.2.1 Principe de génération

L'idée principale est de couvrir le domaine de calcul Ω par des triangles (2D) ou des tétraèdres (3D), en progressant par couches successives d'éléments à l'intérieur du domaine. La frontière entre la partie de Ω qui est maillée et le reste de Ω constitue le front. Ce front est initialisé au début de la procédure de maillage par la frontière du domaine Ω. En 2D, un front est constitué de segments de droite (voir la Figure 1.6),

alors qu'en 3D il est constitué de triangles. Les étapes de la progression de l'algorithme sont les suivantes (François, 1998 ; Golgolab, 1989 ; Löhner et Parikh, 1988):

1. Initialisation du front sur la frontière de Ω.
2. Classement du front selon un critère (taille, qualité... dépend de la dimension de l'espace).
3. Sélection du premier élément du front.
4. Calcul de la position du sommet idéal P.
5. Recherche de la liste des sommets les plus proches existants (nœuds pré-générés ou faisant partie du maillage).
6. Classement des nœuds (selon critère à déterminer).
7. Création d'un élément valide (pas d'intersection avec d'autres éléments, volume positif...) avec le premier sommet qui le permet.
8. Mise à jour du front et reclassement
9. Si le front n'est pas vide (Ω non totalement recouvert), passer à l'étape 3.
10. Étape de lissage et de régularisation du maillage.

Ces étapes sont représentées dans les figures ci-dessous. Elles constituent le canevas d'un grand nombre de méthodes.

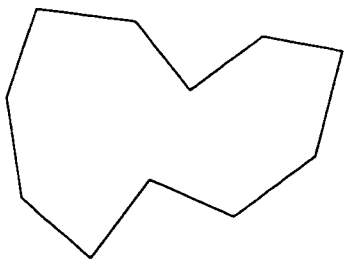


Figure 1.6 : Front initial sur un polygone en 2D

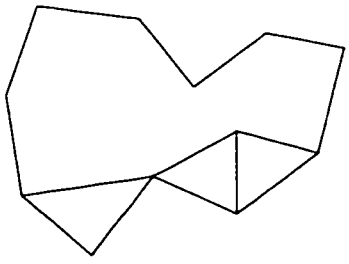


Figure 1.7 : Quelques étapes de l'avance de front

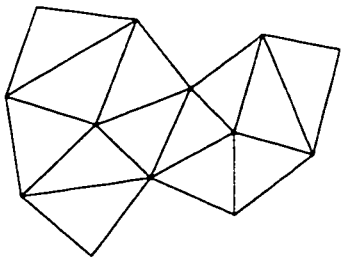


Figure 1.8 : Maillage final. Deux points internes ont été créés

Les différences entre les approches se situent au niveau des heuristiques choisies pour assurer la convergence du mailleur et la génération de simplexes de bonne qualité. Les étapes 2, 4, 6 et 7 de l'algorithme précédent sont implémentées de façons très variables selon les auteurs :

Étape 2 : En 3D, le classement des éléments du front est fait dans l'ordre croissant de leur taille par Rassineux (1995 et 1997), et dans l'ordre croissant de leurs tailles pondérée par leur qualité par Golgolab (1989). En 2D, selon François (1998), il est acquis de classer les segments du front par ordre de longueurs croissantes.

Étape 4 : Les sommets peuvent être générés soit *a priori* (avant la connexion) (Lo, 1991; Rassineux, 1995), soit pendant (Chae 1989; Golgolab, 1989; Cuillière 1998).

Étape 6 : Golgolab (Golgolab, 1989) propose de classer les nœuds du front selon un critère de proximité topologique (voir Figure 1.9) avec l'élément de base.

A : Points adjacents

L : Points liés

V : Points voisins

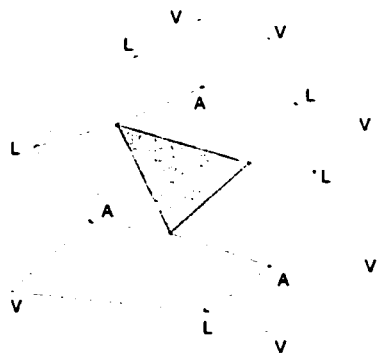


Figure 1.9 : Classification des nœuds selon (Golgolab, 1989)

Étape 7 : Il est nécessaire d'éviter les cas de blocage. En 2D, les blocages sont résolus très simplement (par la réduction de la distance entre le point idéal et le segment

considéré). Il n'existe pas de cas de blocage topologique en 2D. En 3D, la situation est toute autre. Si aucun des nœuds de l'étape 6 ne convient, alors il faut prendre le nœud idéal. Si le nœud idéal ne convient pas, il faut en prendre un autre un peu plus proche du triangle, et recommencer l'opération. Si à l'issue de cette opération, il est toujours impossible de mailler, on a affaire en 3D au cas du polyèdre de Schönhart (1928) (voir la Figure 1.10) et qui constitue un exemple de blocage topologique. Celui-ci n'est en effet pas triangulable dans le cas général sans génération d'un ou plusieurs points internes dits *points de Steiner* (voir la Figure 1.11). Pour contrer ce genre de situation, Golgolab (1989) et Rassineux (1995) proposent de détruire des éléments du maillage, en espérant de se retrouver par la suite dans une configuration non dégénérée. Toutefois, il est clair que cette action peut entraîner le bouclage de l'algorithme (destruction, puis création d'une configuration semblable en un lieu légèrement différent). Ce bouclage est très gênant car il est difficile à détecter. Le mailleur peut en effet faire se déplacer la cavité dans l'ensemble du domaine et ne jamais arriver à la refermer complètement.

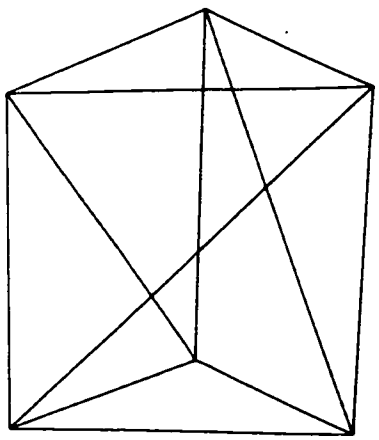


Figure 1.10 : Exemple de polyèdre de Schönhart (non triangulable)

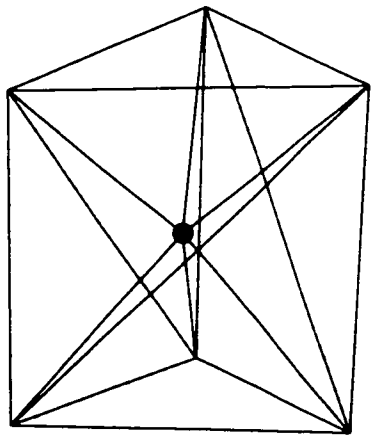


Figure 1.11 : Ajout d'un point de Steiner, décomposition en huit tétraèdres

1.2.3 Autres méthodes

Les autres méthodes de maillage sont principalement les méthodes de décomposition spatiale. Selon les cas, la décomposition est faite en cases de taille régulière (énumération exhaustive de Thacker et al. (1980) qui imposent *de facto* une discrétisation fixe ou selon une décomposition plus apte à saisir les détails géométriques du domaine (octree en 3D, étudié par Yerry et Shephard (1984), quadtree en 2D). Ces méthodes ont l'avantage d'être extrêmement simples à coder, rapides et robustes. Leurs principaux inconvénients sont leur manque de flexibilité quant à la génération de maillages adaptés, et le fait que les éléments générés près des frontières du domaine à mailler sont de mauvaise qualité et souvent trop petits. De plus, il semble très difficile d'adapter ces méthodes à une génération de maillages anisotropes, car les notions de distance et d'orientation ne font pas intimement partie de la méthode de maillage. Ces méthodes reviennent en fait à se ramener à un problème de topologie simple, soit le maillage trivial d'un carré ou d'un cube. Les problèmes sont rencontrés principalement lors du maillage des éléments de la frontière du domaine, où la complexité et le nombre de situations particulières augmentent grandement, et à un point tel qu'en 3D il faut parfois utiliser d'autres méthodes pour effectuer ce maillage.

La méthode de compaction de sphères est en fait un cas particulier de méthode de Delaunay. Elle consiste en un positionnement optimal des sommets, les connexions entre eux (génération de la triangulation du nuage de points à proprement parler) étant effectuées par une méthode de Delaunay par la suite. De par sa nature, elle possède donc certains des inconvénients de la méthode de Delaunay, inconvénients particulièrement critiques en 3D (récupération des frontières du domaine à mailler par exemple). Elle permet toutefois de générer des maillages extrêmement réguliers, sans la nécessité de lissage. Elle permet de plus de générer des maillages anisotropes, au même titre que la méthode de Delaunay (Shimada, 1997).

1.2.3.1 Principe de génération - méthode de quadtree et d'octree

La méthode de quadtree modifiée suit le principe exposé en détail dans les Figure 1.12 *a-h*. De *a* à *f*, on montre la génération de la grille (subdivision de chaque cellule en quatre), étape par étape. En *g*, une subdivision supplémentaire est imposée de façon à ce que deux cellules voisines ne puissent avoir un rapport de taille supérieur à deux. Ceci est effectué pour limiter le nombre de cas différents lors de la transformation des cellules en triangles. En g, on montre précisément le début de cette transformation; elle consiste à paver chaque cellule intérieure selon son type (cellule simple, avec 1 côté découpé, 2 côtés découpés, etc) à l'aide d'un schéma prédéfini (ou patron, template). Les cellules situées sur la frontière sont traitées à part. Si elles font partie d'une des configurations simples (toujours le cas en 2D) alors elles sont maillées de nouveau à l'aide d'un schéma prédéfini. Dans le cas contraire (rencontré parfois en 3D), il faut faire appel à d'autres méthodes de maillage. Enfin, les cellules situées en dehors du domaine sont éliminées.

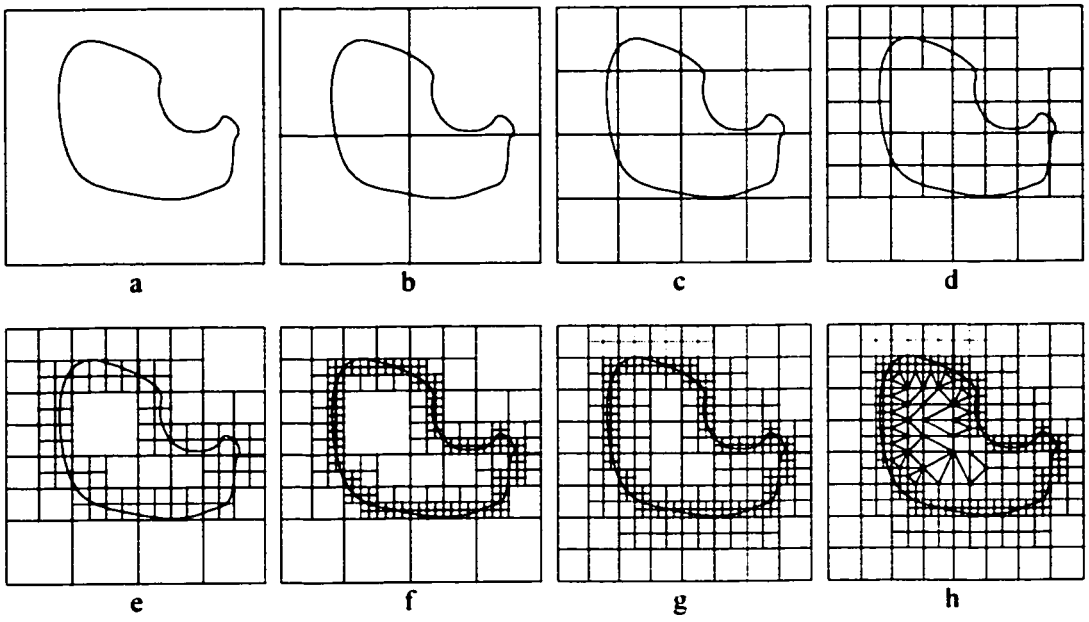


Figure 1.12 : Construction d'un quadtree, puis génération d'un maillage (partiel ici)

1.2.3.2 Principe de génération - méthode de compaction de sphères.

Cette méthode doit son origine à Shimada et Gossard (1993 et 1995). Shimada s'est basé sur des analogies physiques avec des réseaux de bulles, afin de générer un maillage le plus régulier possible. Le fondement de cette méthode revient en fait à générer tous les points du maillage et à les positionner correctement avant de générer la connectivité. En fait cela correspond à placer la phase de lissage (laplacien ou autre) avant de générer les éléments du maillage. Ceci est à opposer aux méthodes précédemment décrites, dans lesquelles les points sont générés le plus souvent en même temps que les éléments, le lissage intervenant à la fin du processus. Les différentes étapes d'un mailleur par compaction de sphères ("sphere packing") sont les suivantes:

1. Détermination de la taille des éléments souhaitée en certains points clefs du domaine Ω.
2. Interpolation sur une grille (régulière ou quadtree et octree) de la carte de taille.

3. Génération de sommets et des sphères. Les sommets sont générés avec une densité en rapport avec la taille de maille locale déterminée à l'étape 2. Le rayon des sphères suit la même règle, i.e. à forte densité, petit rayon et inversement (voir la Figure 1.13).
4. Déplacement des sommets par un processus itératif, en vue d'aboutir à l'équilibre du réseau de bulles. (Figure 1.14 à Figure 1.16). Les bulles sont soumises entre elles à des forces de contact tel qu'indiqué dans la Figure 1.18. Éventuellement des bulles sont ajoutées ou éliminées.
5. Création du maillage à partir du réseau de points à l'aide d'une méthode de Delaunay contrainte (voir la Figure 1.17)

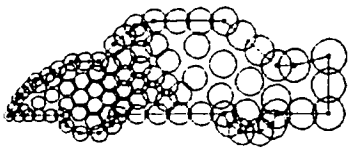


Figure 1.13: Génération des bulles et des sommets initiaux

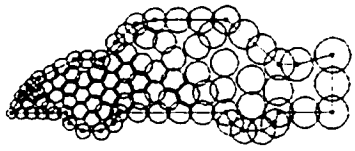


Figure 1.14: Première itération du processus de "sphere packing"

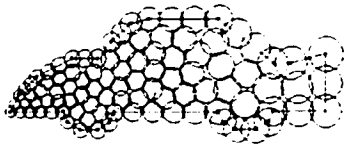


Figure 1.15: Deuxième itération

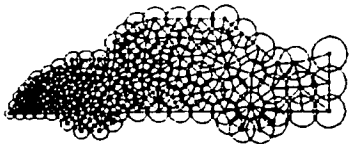


Figure 1.16: Sixième itération. La triangulation de Delaunay est indiquée

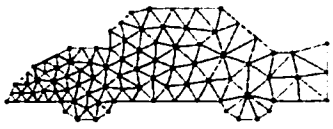


Figure 1.17: Triangulation finale

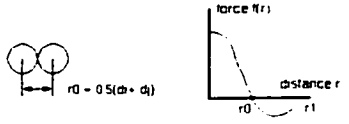


Figure 1.18: Force exercée entre deux bulles en fonction de leur distance. r0 correspond à l'état d'équilibre

1.3 Adaptation anisotrope

Les simulations numériques impliquent souvent des géométries courbes (surface d'un solide par exemple), et certaines méthodes de maillage s'appliquent à partir de la représentation paramétrique des surfaces. Il convient donc de faire en sorte que le maillage dans l'espace réel possède les caractéristiques voulues. La transformation qui fait passer de l'espace paramétrique à l'espace réel ne conserve pas, dans la plupart des cas, les angles et les distances. Un maillage isotrope dans l'espace de référence peut donc très bien ne plus l'être une fois transporté dans l'espace réel (Cuillière, 1998; Tristano, 1998). On peut pallier à cet inconvénient en maillant judicieusement de façon anisotrope dans l'espace paramétrique pour obtenir un maillage isotrope dans l'espace réel. C'est une première raison, géométrique, qui justifie les maillages anisotropes, même si l'on désire obtenir un maillage isotrope dans l'espace réel. Ceci concerne principalement le maillages des surfaces paramétriques de type CAO.

Dans le cadre des calculs d'écoulements, des phénomènes très directionnels comme les couches limites, les chocs et les déplacements de fronts de matière sont courants. Afin de traiter correctement ces phénomènes, les maillages doivent posséder des caractéristiques d'anisotropie particulières, aptes à rendre compte des variations rapides de la solution dans ces zones (Vallet, 1992). Ces caractéristiques sont dictées principalement par un calcul d'erreur (matrice Hessienne d'un champ solution dans la plupart des cas). Ceci constitue la seconde raison d'être des maillages anisotropes : de leur adéquation dépend la précision de la résolution du problème physique. Dans ce cadre, les perspectives d'application des maillages anisotropes sont vastes, aussi bien en 2D qu'en 3D.

1.3.1 Notion de métrique

De façon à pouvoir générer un maillage anisotrope, il est nécessaire de redéfinir la notion de distance dans l'espace considéré. Ceci est généralement effectué en considérant la notion de métrique. La métrique est définie en tout point P du domaine d'étude Ω, et elle n'est généralement pas constante. Toutefois, si elle est continue, cela définit un espace riemannien (Schutz, 1980). Elle est représentée par une matrice $d \times d$ (notée **M** dans la suite). Elle est symétrique, définie positive et dépend de la position du point P dans le cas général. En deux dimensions, la matrice prend la forme suivante :

$$\mathbf{M}(P) = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \text{ telle que } (ac - b^2) > 0; a > 0; c > 0. \tag{4}$$

La notion de distance est définie à l'aide de cette métrique par une intégrale selon une géodésique Γ parcourue pour aller d'un point A à un point B :

$$dist(AB) = l(\Gamma) = \int_0^1 \sqrt{{}^t\mathbf{s}'(t) \cdot \mathbf{M}(s(t)) \cdot \mathbf{s}'(t)}\, dt \tag{5}$$

où $\mathbf{s}(t)$ représente une paramètrisation du chemin telle que $\mathbf{s}(0) = \overrightarrow{OA}$ et $\mathbf{s}(1) = \overrightarrow{OB}$. La notation $\mathbf{s}'(t)$ représente le vecteur tangent à ce chemin. Il existe une infinité de chemins pour aller d'un point A à un point B, mais la distance entre deux points correspond au plus court chemin dans la métrique considérée: la géodésique. Ce chemin particulier est difficile à déterminer dans le cas général, ce qui impose de se restreindre en considérant des métriques localement constantes (métrique du plan tangent). On peut prendre comme référence un point G qui se trouve dans le voisinage de A et de B. Dans ce cas,

les géodésiques sont des droites, localement l'espace est euclidien, et la distance est alors définie dans le voisinage de G selon la formule:

$$dist_G(AB) = \sqrt{{}^t\overrightarrow{AB} \cdot \mathbf{M}(G) \cdot \overrightarrow{AB}} \tag{6}$$

car il est maintenant possible d'intégrer l'expression (5).

De la même façon, on peut définir un produit scalaire et sa norme associée, entre deux vecteurs ***p*** et ***q*** :

$$\langle \mathbf{p}, \mathbf{q} \rangle_G = {}^t\mathbf{p} \cdot \mathbf{M}(G) \cdot \mathbf{q} \tag{7}$$

$$\|\mathbf{p}\|_G = \sqrt{\langle \mathbf{p}, \mathbf{p} \rangle_G} \tag{8}$$

Ceci permet de redéfinir de façon tout à fait classique la notion d'angle entre deux vecteurs:

$$\cos\theta|_G = \cos(\mathbf{p}, \mathbf{q})|_G = \frac{\langle \mathbf{p}, \mathbf{q} \rangle_G}{\|\mathbf{p}\|_G \cdot \|\mathbf{q}\|_G} \tag{9}$$

Dans le cas de l'anisotropie géométrique, le tenseur métrique peut être trouvé à partir des relations suivantes en fonction des relations entre les coordonnées (x, y, z) de l'espace réel et les paramètres (u, v) de l'espace paramétrique (première forme fondamentale):

$$a = \left(\frac{\partial x}{\partial u}\right)^2 + \left(\frac{\partial y}{\partial u}\right)^2 + \left(\frac{\partial z}{\partial u}\right)^2 \tag{10}$$

$$b=\left(\frac{\partial x}{\partial v}\right)^2+\left(\frac{\partial y}{\partial v}\right)^2+\left(\frac{\partial z}{\partial v}\right)^2 \tag{11}$$

$$c=\frac{\partial x}{\partial u}\frac{\partial x}{\partial v}+\frac{\partial y}{\partial u}\frac{\partial y}{\partial v}+\frac{\partial y}{\partial u}\frac{\partial y}{\partial v} \tag{12}$$

1.3.2 Adaptation du mailleur de Delaunay

En utilisant les notions de métriques anisotropes, il est possible selon (Borouchaki et al., 1997; George et Borouchaki, 1997; Remacle, 1999) de construire un mailleur de Delaunay qui respecte une métrique anisotrope, tout en conservant l'architecture générale du noyau de Delaunay vue à la section § 1.2.1.2. Il faut néanmoins redéfinir la mesure de Delaunay qui dépend de la façon dont on mesure les distances :

$$\alpha_G(P,K)=\frac{dist_G(P,O_K)}{r_K} \tag{13}$$

$$\text{avec } r_K=dist_G(O_K,K_i) \tag{14}$$

où K_i est un des $d+1$ sommets du simplexe K et O_K le centre du cercle circonscrit (cercle au sens de la métrique considérée, c'est à dire un ellipsoïde dans l'espace réel).

Le point faible de cette méthode est l'approximation de la métrique au moment de la détermination des simplexes à éliminer, en particulier au début de la procédure d'insertion car les simplexes traversent en effet tout le domaine. Il existe plusieurs possibilités pour approcher cette métrique. On peut par exemple considérer la métrique au point que l'on veut insérer dans le maillage, et la supposer bonne pour tout le

voisinage (Remacle, 1999). La condition d'acceptation du triangle dans la cavité définie au §1.2.1.2 reste valable *dans la métrique du point P*:

$$\alpha_P(P,K)<1 \tag{15}$$

Ceci présente clairement des difficultés au début du processus d'insertion. Une bonne méthode (Borouchaki et al., 1997) semble être de considérer la métrique de deux points, en l'occurrence le point P à insérer et le point P_K correspondant à l'arête qui serait formée avec P si le triangle à tester était retenu. Ceci implique une autre condition d'acceptation du triangle dans la cavité :

$$\alpha_{P_K}(P,K)+\alpha_P(P,K)<2 \tag{16}$$

Cette dernière condition est judicieuse puisqu'elle permet d'affirmer que les arrêtes créées par étoilement avec le point P seront intrinsèquement acceptables, à la condition que la métrique soit monotone entre les deux extrémités. Ceci impose certaines restrictions sur les variations de la densité du maillage.

Cette adaptation est faite en ne modifiant quasiment pas le noyau de Delaunay, tout simplement en considérant que le maillage à générer est isotrope et de taille normalisée à 1. C'est le champ de métrique qui permet de faire les calculs de distance.

1.3.3 Adaptation du mailleur frontal

Très peu de travaux traitent de l'adaptation anisotrope des mailleurs frontaux. On peut citer Cuillère (1998) et Tristano (1998) qui réussissent à générer des maillages sur des surfaces paramétriques. Le maillage anisotrope est donc effectué dans le plan. A notre

connaissance, en trois dimensions, il n'existe pas de mailleur frontal commercial capable de suivre une carte d'anisotropie. Seul Golgolab (1989) propose "d'étirer" les éléments près des frontières et de tenir compte des chocs dans le cadre d'un calcul d'aérodynamique. Le problème principal rencontré dans la génération de maillages anisotropes ou adaptés en taille avec une méthode frontale semble être la difficulté de faire converger l'algorithme, comme le souligne François (François, 1998) pour le suivi d'une carte de taille isotrope.

Cuillière et Tristano proposent de traiter le cas de l'anisotropie géométrique en 2D en reformulant la position du sommet idéal dans la métrique considérée. En 2D isotrope, le sommet idéal doit être généré sur la médiatrice du segment du front considéré, i.e., à angle droit. C'est aussi le cas lorsque l'on a affaire à une paramétrisation isotherme d'une surface. Dans la paramétrisation de surfaces gauches, cette direction n'est plus, dans le cas général, selon un angle droit. Il faut donc trouver l'angle θ selon lequel générer le sommet. Si les coordonnées des extrémités du segment considéré sont $A(u_1, v_1)$ et $B(u_2, v_2)$ dans l'espace paramétrique (voir la Figure 1.19), cet angle est solution de :

$$\tan(\theta) = \frac{(a-b)(u_2-u_1)(v_2-v_1) - c\left((u_2-u_1)^2 - (v_2-v_1)^2\right)}{a(u_2-u_1)^2 + b(v_2-v_1)^2 + 2c(u_2-u_1)(v_2-v_1)} \qquad (17)$$

avec les notations du §1.3.1 concernant les composantes a, b et c du tenseur métrique. De plus, comme dans le cas du mailleur de Delaunay, il faut calculer des distances selon une expression de la forme de l'intégrale (5). On approche cette distance directement en suivant une droite dans l'espace paramétrique. Cela ne correspond pas dans le cas général à la distance la plus courte dans l'espace réel.

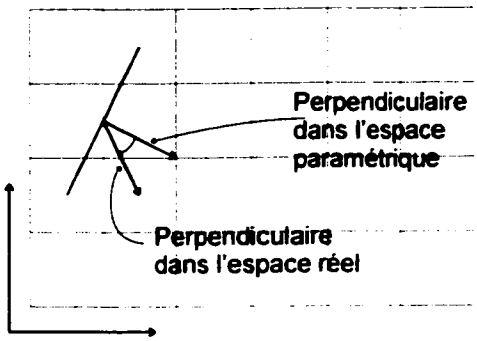


Figure 1.19 : Distorsion dans l'espace paramétrique

Un angle de décalage θ est calculé de façon semblable, et les calculs de distance entre deux points A et B sont effectués à l'aide de la moyenne des distance mesurées par rapport à la métrique en chacun des deux points si la surface est suffisamment régulière :

$$dist_{avg}\left(\overrightarrow{AB}\right)=\frac{dist_{A}\left(\overrightarrow{AB}\right)+dist_{B}\left(\overrightarrow{AB}\right)}{2} \tag{18}$$

Si la surface ne l'est pas, il faut effectuer une intégration numérique le long du segment de droite (AB), en utilisant n points d'intégration $\{P_1 \ldots P_n\}$. Ce segment n'est toutefois pas une géodésique, comme indiqué plus haut:

$$dist_{avg}\left(\overrightarrow{AB}\right)=\sum_{i=1}^{n} dist_{i} \tag{19}$$

$$dist_{i}=dist_{avg}\left(\overrightarrow{P_{i}P_{i+1}}\right) \tag{20}$$

Curieusement, cette façon de procéder ne tire pas parti de toutes les généralisations qui peuvent être induites par l'utilisation des tenseurs métriques. En effet, la génération du maillage reste anisotrope. Nulle part il n'est utilisé d'espace "de référence" dans lequel le maillage serait isotrope et de taille fixe. Pour pouvoir imposer au maillage de respecter une carte d'anisotropie, c'est pourtant la solution la plus élégante car elle n'impose pas le calcul d'expression compliquée concernant l'angle θ. Tout se fait naturellement par le changement de métrique. Somme toute, si l'on devait générer un maillage anisotrope dans un domaine plan, on ne se servirait de la métrique que comme guide pour la création des mailles, l'algorithme de maillage étant anisotrope et donc différent de celui utilisé auparavant pour générer un maillage isotrope.

1.3.4 Adaptation des autres mailleurs

À notre connaissance, il n'existe de mailleurs par décompositions spatiale ou quadtree et octree capables de traiter des cas anisotropes. L'adaptation du mailleur par compaction de sphères existe, pour le moment uniquement en 2D (Shimada, 1997). Le principe consiste à utiliser des ellipsoïdes en lieu et place des sphères, et de relier les centres des ellipsoïdes pour former le maillage final par une méthode de Delaunay anisotrope, comme décrite plus haut. Bien entendu la formule schématisée sur la Figure 1.18 doit être modifiée car les relations entre les différents "bulles" (ellipsoïdes) impliquent des forces (voir la Figure 1.18) qui ne sont plus indépendantes de leur orientation relative. Ceci mis à part, le schéma de construction est rigoureusement le même que dans le cas isotrope (voir Figure 1.20 à Figure 1.23)

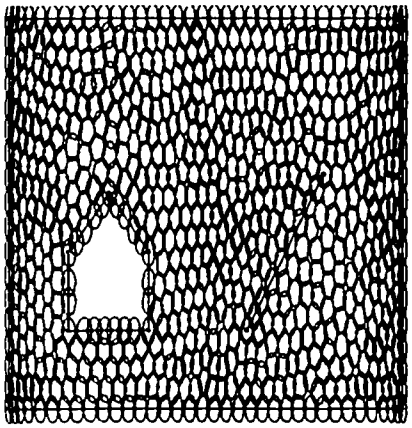


Figure 1.20: Bulles dans l'espace paramétrique

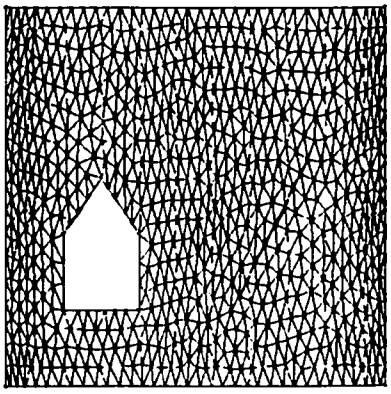


Figure 1.21: Triangulation dans l'espace paramétrique

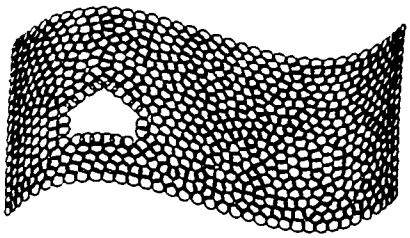


Figure 1.22: Bulles dans l'espace réel.

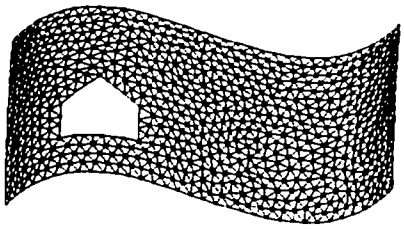


Figure 1.23: Triangulation dans l'espace réel.

1.4 Suivi de front

La génération de maillages pour des problèmes à frontière et / ou front mobile a fait l'objet de peu de résultats. Il existe deux approches. Il est à noter que le choix de l'une ou de l'autre n'est pas dénué de conséquences pour l'aspect de la génération de maillages.

La plus courante consiste en une discrétisation de type éléments finis pour l'espace, alors que le temps est discrétisé selon un schéma de différences finies. Dans cette voie, on peut citer Hassan et al. (1998) qui traite du problème de l'adaptation de maillages pour

les calculs de fluides compressibles lors de modifications de la géométrie (pièces en mouvement). Si la géométrie est peu changeante (faibles oscillations autour d'une position moyenne par exemple), la modification du maillage est faite par un simple déplacement de points et lissage (l'analogie avec un réseau de ressorts est employée). Dans le cas de mouvements de grande amplitude ou de changements de topologie, il est nécessaire de remailler certaines zones autour des frontières. Hassan effectue cette opération par ajout et retrait automatique de points selon une méthode de Delaunay. Le problème est la récupération de la frontière de la zone remaillée qui n'est absolument pas évidente avec une méthode de Delaunay, en particulier en 3D. Hassan précise qu'il est nécessaire de récupérer exactement les frontières du sous-domaine remaillé, de façon à éviter les complications lors de sa connexion au reste du maillage. La méthode proposée n'est pas implémentée pour les maillages anisotropes, mais l'adaptation semble ne pas poser de problèmes particuliers. Li (1998) propose un algorithme basé sur le principe de construction d'un maillage régulier considérant un empilement compact de sphères. L'idée principale de Li est de proposer un algorithme général permettant de traiter le cas des maillages évolutifs, cas plus général que les problèmes de front mobile. Pour ce faire, il propose de traiter simultanément le cas du raffinement (par exemple en amont d'un front), et le cas du déraffinement (en aval du front). La force de son approche est qu'il contrôle de façon rigoureuse la position des sommets de façon à générer un maillage respectant au mieux une carte de taille changeante. La connexion des points est faite par une méthode de Delaunay. Cet algorithme n'est implémenté que pour des maillages isotropes, bien qu'il existe une version anisotrope de la méthode de compaction de sphères (Shimada et Gossard, 1995).

La seconde approche est l'utilisation d'éléments finis espace-temps. Ceci implique de mailler le domaine constitué par les d dimensions d'espace, plus la dimension temporelle. Dans le cas de simulations en trois dimensions (spatiales), le support de calcul aura quatre dimensions. Ceci pose un problème de maillage évident, étant donné qu'il n'existe pas actuellement de mailleur opérant en quatre dimensions. Un approche

adaptée aux volumes finis (et uniquement en deux dimensions spatiales) est proposée par Zwart et Raithby (1998). Un inconvénient de cette méthode est la nécessité de redévelopper tout un ensemble d'outils d'adaptation de maillage, car il n'est absolument pas assuré que le premier maillage généré permette d'obtenir des résultats précis. Comme dans le cas de la simulation d'un problème stationnaire, il est parfois nécessaire d'effectuer plusieurs fois la boucle mailleur-résoluteur afin d'obtenir un bon résultat. De plus, il faut modifier la formulation des problèmes qui étaient auparavant résolus à l'aide de méthodes classiques découplées du temps. Il existe quelques travaux sur l'adaptation de maillages espace-temps pour l'équation des ondes (Collino, 1998).

1.5 Procédé d'injection de résine sur renforts

Le procédé d'injection de résine sur renfort est utilisé pour produire des pièces composites à une cadence soutenue comparativement aux autres méthodes telles que l'imprégnation manuelle ou l'utilisation de prépregs. En contrepartie, les performances mécaniques ne sont évidemment pas du même ordre. La principale difficulté est la prédiction du positionnement des points d'injection et des évents, de façon à éviter la présence de zones sèches. La phase de cuisson (polymérisation) a aussi une grande influence sur la qualité des pièces obtenues par ce procédé. La Figure 1.24 schématise le procédé. On peut référer à (Cauchois, 1997; Rudd et al., 1997) pour une description exhaustive.

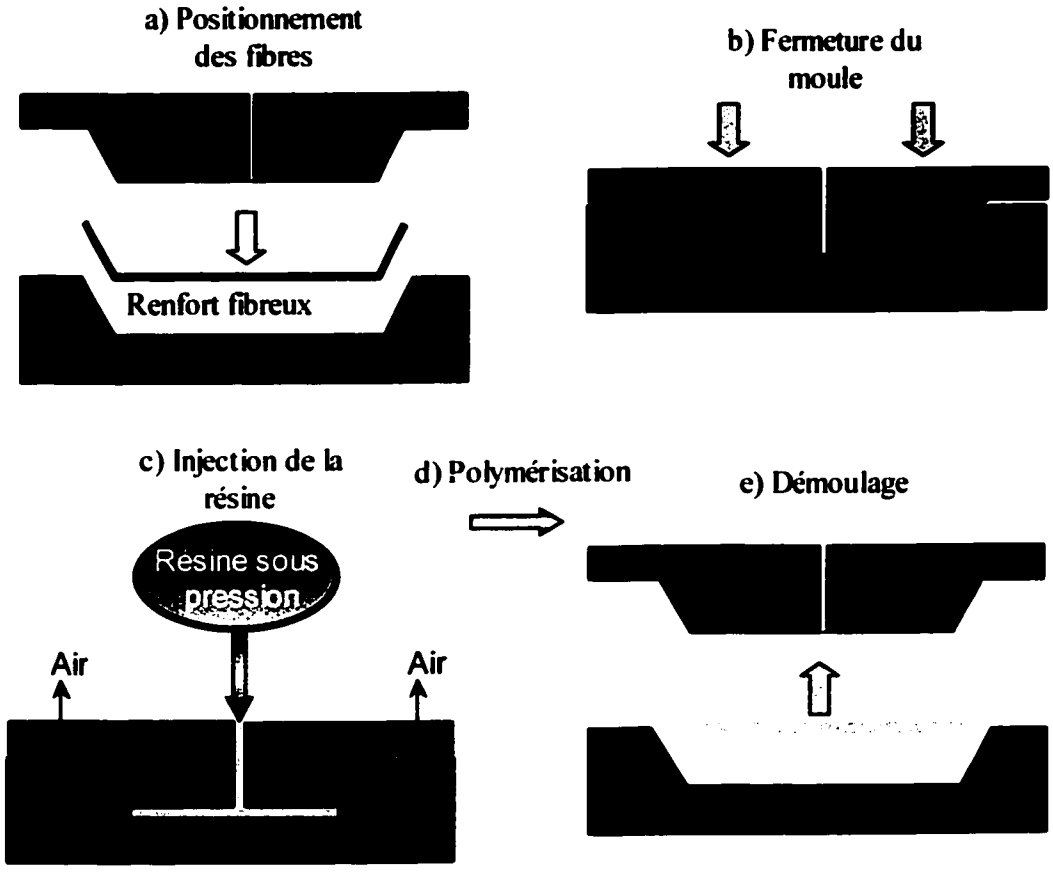


Figure 1.24 : Procédé RTM

1.6 Conclusion

On peut constater dans cette étude bibliographique qu'un grand nombre de travaux existent dans le cadre de la génération de maillages non structurés et isotropes. Ces travaux s'appuient sur des méthodes diverses, dont les plus importantes sont la méthode d'avance de front et la méthode de Delaunay. Ces dernières offrent une souplesse incomparable aux (quelques) autres méthodes. Toutefois, les travaux impliquant des maillages anisotropes sont plus rares et relativement récents. De même, pour ce qui a trait à l'adaptation de maillages dans le cadre de frontières mobiles, les travaux sont encore plus récents et partiels. Ceux ci dépendent souvent très fortement de l'application

envisagée. Notre recherche implique ces deux aspects pour l'application au moulage par injection de résine sur renfort. A notre connaissance, il n'existe pas encore de travaux dans ce domaine.

CHAPITRE 2
ORGANISATION GÉNÉRALE

La thèse présentée ici est sous la forme d'une thèse par articles. Quatre articles la composent et sont reliés logiquement entre eux. Les résultats et discussions des travaux réalisés dans cette étude sont présentés en détail dans les articles. Ces articles sont présentés aux chapitres trois, quatre, cinq et six et ont été soumis pour publication dans des revues internationales reconnues dans leurs domaines respectifs (modélisation géométrique pour le premier article, modélisation numérique des matériaux composites pour les trois suivants). Au moment où ce texte est rédigé, le premier article est déjà publié, le second est accepté est sera publié dans les mois qui suivent, et les deux derniers sont soumis et en processus actif de révision. Afin de faire ressortir la complémentarité des articles, nous présentons brièvement une synthèse de la recherche proposée, puis à la suite de celle ci, nous présentons d'un manière intégrée les quatre articles.

2.1 Recherche proposée

La recherche proposée ici concerne la génération de maillage pour améliorer les simulations numériques d'un procédé de fabrication de structures en matériaux composites. Ce procédé est appelé RTM pour Resin Transfer Moulding, et consiste en l'injection de résine dans un moule fermé contenant un renfort poreux composé de tissus de fibres de carbone ou de verre. Il s'agit donc d'un écoulement à frontière libre obéissant à la loi de Darcy des milieux poreux saturés. L'objectif de la thèse est de proposer un algorithme de remaillage visant à rendre la simulation des écoulements de Darcy dans les renforts pour matériaux composites plus précise et plus rapide que ce qui est actuellement fait sur une maillage fixe et isotrope.

2.2 Maillage surfacique (article 1)

Dans un premier temps, une méthode de génération de maillage surfaciques est proposée, supposée connue une triangulation particulière de la surface (sous forme de fichier STL). Cette contribution vise à montrer la possibilité de générer un maillage surfacique respectant une carte de taille (isotrope dans un premier temps) sans connaître la surface réelle (celle ci est définie auparavant dans un module de CAO, mais inaccessible pour diverses raisons). De nombreux exemples montrent que les algorithmes proposés fonctionnent bien. Le lien avec le sujet de la thèse est le suivant : dans le cadre de la simulation d'injection sur renfort, on dispose comme données de base d'un maillage surfacique et de conditions d'injection. D'une façon ou d'une autre, les conditions d'injections et la position du front de résine déterminent la carte de taille d'un maillage désirable pour poursuivre la simulation. Cette partie traite donc précisément de ce cas.

2.2.1 Présentation

Le Chapitre 3 est consacré à la génération de maillages surfaciques à partir de fichiers de CAO usuellement dédiés à la stéréo-lithographie. En fait, ces fichiers (fichiers « STL ») sont constitués d'une triangulation de la surface d'une géométrie solide. Toutefois, celle ci est basée uniquement sur des critères d'approximation géométriques. Ceci exclut toute utilisation en l'état pour effectuer des calculs par éléments finis. En particulier, les éléments peuvent être extrêmement allongés selon les courbures principales de la surface de l'objet. L'intérêt de l'approche est de pouvoir utiliser n'importe quel modeleur solide pour générer un maillage qui soit directement utilisable pour des calculs numériques de coques. Dans ce cas, les éléments du maillage doivent respecter un facteur de forme compatible avec un calcul par éléments finis. Il est aussi possible de se servir de ce maillage comme point de départ pour générer un maillage volumique à l'aide d'un autre programme. Toutefois, ce dernier requiert une triangulation de la surface du solide qui

respecte les mêmes contraintes que précédemment, car le facteur de forme des éléments volumiques généré dépend fortement de celui de la surface, au moins dans son voisinage immédiat.

Cet article a été soumis, accepté et publié dans la revue « Computer Aided Design ».

É. Béchet, J.C. Cuillière, F. Trochu, 2002, Generation of a finite element mesh from stereolithography (STL) files, Computer Aided Design **34**, pp.1-17.

2.3 Suivi de front, maillage adaptatif et anisotrope (article 2)

Dans cet article, les algorithmes présentés précédemment sont repris et une extension de la méthode de maillage est proposée pour les cartes de tailles anisotropes arbitraires. Ensuite, une carte de taille anisotrope adaptée au problème de l'écoulement est construite pour chaque pas de temps de la simulation numérique. Cette carte de taille définit un nouveau maillage, obtenu par l'utilisation des méthodes de maillage décrites précédemment. Un algorithme de progression de front par *level-set* est implémenté, et permet de faire évoluer la position de la frontière libre en fonction de la résolution du problème (elliptique) en pression. Dans le cas présent, il n'y a pas de phénomènes de transport, il s'agit juste de faire évoluer une surface libre. Ainsi, comme le montre la validation menée dans l'article il est possible de réduire considérablement le nombre de pas de temps de la simulation en orientant préférentiellement les éléments du maillage, et en produisant pour chaque pas de temps un maillage adapté en taille.

2.3.1 Présentation

Le Chapitre 3 a montré la possibilité de modifier dans de grandes proportions un maillage surfacique sans connaître la base géométrique (CAO) dont il est issu. Dans le

Chapitre 4, ces aspects sont enrichis par la prise en compte de cartes de taille anisotropes afin de pouvoir traiter des problèmes exhibant des directions privilégiées (aérodynamique, équations de transport). Ensuite, le couplage avec un solveur est effectué afin de générer des maillages anisotropes et adaptatifs pour résoudre un problème d'écoulements à surface libre en milieux poreux. Des études précédentes on montré l'utilité d'utiliser des éléments linéaires non conformes; c'est dans ce cadre que cette recherche s'inscrit. L'objectif est de permettre de raffiner spécifiquement le maillage dans le voisinage de la surface du fluide. Deux buts sont poursuivis : améliorer la précision dans l'évolution de la surface libre, et permettre une amélioration de la précision des simulations thermiques. Ces dernières sont en effet très dépendantes de la discrétisation au voisinage de la surface libre. Une approche par *level-set* pour l'évolution de la surface libre est avantageusement couplée à l'algorithme de génération de maillage. Les résultats obtenus sont ensuite comparés avec des résultats d'expérience et de simulation provenant d'un logiciel de calcul spécialisé pour fins de validation.

Cet article a été soumis, accepté et sera publié prochainement (~janvier 2003) dans la revue « Journal of Reinforced Plastics and Composites ».

2.4 Phénomènes de transport thermique et génération d'un maillage optimal partie 1 – cas bidimensionnel (article 3)

Dans un premier temps, les algorithmes de l'article 2 sont adaptés aux phénomènes de transport thermique (transport et diffusion thermique, et transport du taux de conversion de la résine). On montre qu'il n'est pas possible de conserver de grands pas de temps à cause de la condition sur le nombre de Courant pour les méthodes Euleriennes. Une adaptation de l'algorithme est proposée qui, spécifiant un pas de temps différent pour la résolution de l'avancée de la surface libre et la résolution des équations de transport, permet d'assurer la stabilité du schéma numérique. Par la suite, un algorithme pour générer un maillage optimisé *a priori* est proposé. Ce maillage doit borner l'erreur

d'interpolation en pression (afin d'assurer une évolution précise de la surface libre dans le temps), et assurer la condition de Courant pour la stabilité du schéma associé aux équations de transport. On montre théoriquement que ces deux condition peuvent être antagonistes pour un cas particuliers d'injection centrale.

2.4.1 Présentation

Le Chapitre 5 est constitué de trois volets. Une première partie montre la possibilité d'effectuer de l'adaptation *a priori* en vue d'éviter le remaillage durant la simulation proprement dite. Ceci implique le développement d'un estimateur d'erreur qui tienne compte à la fois de l'aspect diffusif de l'équation de Darcy (variable de pression), et de l'aspect convectif. En somme, un maillage intégrant toutes ces caractéristiques est généré et utilisé tout au long d'une simulation. Une validation bidimensionnelle est présentée, sur une géométrie identique à celle du Chapitre 4. La seconde partie de ce chapitre montre l'adaptation de l'algorithme de remaillage du Chapitre 4 aux phénomènes thermiques, en particulier au problème de transport-diffusion. La nécessité de tenir compte de la condition CFL est mise en avant, et une modification de l'algorithme de simulation est proposée. Enfin, des résultats de simulation montrant que l'on réduit la diffusion au voisinage de la surface libre sont présentés. La troisième partie est une analyse théorique d'un cas d'injection centrale. Le but est de montrer qu'il peut être difficile de générer un maillage unique pour résoudre de façon optimale un problème de convection, car les critères de raffinement sont antagonistes selon que l'on considère un maillage uniformisant l'erreur d'interpolation en pression ou un maillage optimal pour résoudre le problème de transport.

Cet article a été soumis à la revue « Composites Part A: Applied Science and Manufacturing ».

2.5 Phénomènes de transport thermique et génération d'un maillage optimal partie 2 – estimateur d'erreur adapté aux surfaces discrètes et extension aux surfaces courbes (article 4)

La procédure présentée dans l'article 3 est étendue aux cas des surfaces gauches. Dans la recherche menée ici, les surfaces considérées sont discrètes et la surface de référence définie en CAO n'est pas accessible. Un estimateur d'erreur est donc dérivé, qui tient compte de l'aspect discret des surfaces en découplant l'erreur purement fonctionnelle de l'erreur géométrique commise lors de la discrétisation du domaine. Enfin, une simulation sur une pièce réelle montre la validité de l'approche.

2.5.1 Présentation

Le Chapitre 6 est la suite directe du Chapitre 5. Il s'agit d'une adaptation aux surfaces courbes de l'estimateur d'erreur présenté dans le Chapitre 5. Les estimateurs d'erreur ne sont en général utilisables que si la géométrie est lisse et que l'on connaît exactement le modèle géométrique sous jacent. En effet, l'erreur d'interpolation intègre l'erreur fonctionnelle au sens des éléments finis ainsi que l'erreur géométrique induite par l'aspect discret des éléments utilisés (en général linéaires). Dans le cas de surfaces discrètes, cela induit un artifice : le raffinement aux alentours d'arrêtes inexistantes dans le modèle réel, mais présentes dans la version discrète de la géométrie. Le but ici est de montrer que l'on peut s'affranchir de ces limites et utiliser un estimateur adapté qui ne tient compte que de l'erreur fonctionnelle. Une application sur une pièce industrielle est présentée.

Cet article a été soumis à la suite de l'article du Chapitre 5 dans la revue « Composites Part A: Applied Science and Manufacturing »

CHAPITRE 3
GENERATION OF A FINITE ELEMENT MESH FROM STEREOLITHOGRAPHY (STL) FILES

E. BÉCHET[1,2], J.-C. CUILLIERE[2], F. TROCHU[1]

(1) Centre de Recherches Appliquées Sur les Polymères (CRASP), Département de Génie Mécanique, École Polytechnique de Montréal, H3C 3A7, Canada

(2) Laboratoire de productique, Département de Génie Mécanique, Université du Québec à Trois-Rivières, G9A 5H7, Canada

Email: bechet@meca.polymtl.ca

3.1 Abstract

The aim of the method proposed here is to show the possibility of generating adaptive surface meshes suitable for the finite element method, directly from an approximated boundary representation of an object created with CAD software. First, we describe the boundary representation, which is composed of a simple triangulation of the surface of the object. Then we will show how to obtain a conforming size-adapted mesh. The size adaptation is made considering geometrical approximation and with respect to an isotropic size map provided by an error estimator. The mesh can be used "as is" for a finite element computation (with shell elements), or can be used as a surface mesh to initiate a volume meshing algorithm (Delaunay or advancing front). The principle used to generate the mesh is based on the Delaunay method, which is associated with refinement algorithms, and smoothing. Finally, we will show that not using the parametric representation of the geometrical model allows us to override some of the limitations of conventional meshing software that is based on an exact representation of the geometry.

Keywords: Mesh generation, STL File format, Bisection algorithm, Delaunay triangulation.

3.2 Interface with CAD

3.2.1 CAD-based data sets

Most CAD software on the market can generate STL files, and these are generally used for prototyping and rendering purposes. These files represent a triangulation boundary of the solid. Algorithms for the generation of STL triangulation are highly efficient, and the surface can be approximated very precisely if very large data sets are accepted. However, this is not an accurate representation of the real geometry model because the STL file format is composed only of an extensive list of triangle facets. These facets are composed of the coordinates of the 3 vertices of the triangle, in addition to the coordinates of the normal oriented to the exterior of the solid. This triangulation is built to minimize a geometric approximation criterion that is related to the real boundary of the solid.

```
Solid AutoCAD
   facet normal 0.0000000e+000 0.0000000e+000 1.0000000e+000
      outer loop
A        vertex 1.0000000e+001 0.0000000e+000 1.0000000e+001
B        vertex 1.0000000e+001 1.0000000e+001 1.0000000e+001
C        vertex 0.0000000e+000 1.0000000e+001 1.0000000e+001
      endloop
   endfacet
   facet normal 0.0000000e+000 0.0000000e+000 1.0000000e+000
      outer loop
A        vertex 1.0000000e+001 0.0000000e+000 1.0000000e+001
C        vertex 0.0000000e+000 1.0000000e+001 1.0000000e+001
D        vertex 0.0000000e+000 0.0000000e+000 1.0000000e+001
      endloop
   endfacet
...
   facet normal 1.0000000e+000 0.0000000e+000 0.0000000e+000
      outer loop
E        vertex 1.0000000e+001 1.0000000e+001 0.0000000e+000
B        vertex 1.0000000e+001 1.0000000e+001 1.0000000e+001
F        vertex 1.0000000e+001 0.0000000e+000 0.0000000e+000
      endloop
   endfacet
endsolid AutoCAD
```

Figure 3.1: Example of an ASCII STL file (there is also a binary file format, which contains the same information). Please note the redundancy of vertices A, B and C

3.2.2 Characteristics of STL triangulation

STL triangulation cannot be used directly in the finite element method (FEM), mainly because it requires specific characteristics in the geometrical description of the domain to be computed. In FEM, the geometric and functional support is provided by the elements (triangles or others), and they must have a specific shape, i.e. the proper size and the proper quality factor for the error estimator of the calculation to be as low as desired. In this work, we used the following quality factor for triangles, where d_i , $i = 0..2$ represents the length of each side of the triangle. We assume:

$$d = \max_{i=0..2}(d_i) \quad \text{and} \quad p = \frac{\sum_{i=0}^{2} d_i}{2}$$

The quality factor is then:

$$q = \frac{\sqrt{12}}{d} \cdot \sqrt{\frac{\prod_{i=0}^{2}(p - d_i)}{p}} \tag{21}$$

This quality factor is bounded between 0 for all kinds of degenerated triangles and 1 for equilateral triangles. Of course, these requirements are related to the application. For example, in fluid mechanics it is very common to use anisotropic elements in boundary layers and/or shocks. It is clear that a mesh obtained by minimization of a geometric criterion cannot fit FEM requirements because, depending on the curvatures and topology of the surface, the triangles generated can be greatly stretched in a certain direction. (see Figure 3.2 & Figure 3.3). However, the mesh obtained is generally conforming[1].

---

[1] STL files are theoretically conforming even if this is not always the case in practice. However, if the solid to be meshed is designed properly (no overlapping between surfaces, no gaps, holes etc.), the STL file is usually conforming, and therefore can be used "as is". In this paper we always consider conforming STL triangulations as input.

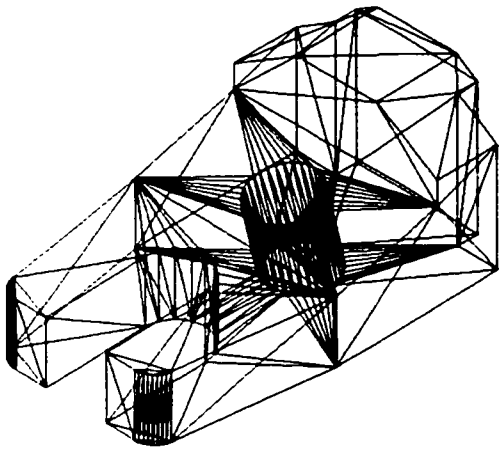


Figure 3.2: "Wire frame" representation of a STL mesh. Notice the stretched triangles near the hole

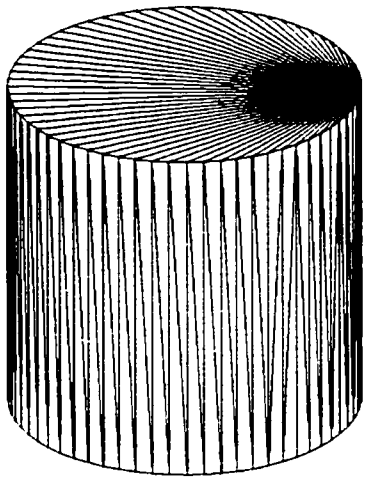


Figure 3.3: Hidden face representation of the STL mesh of a cylinder

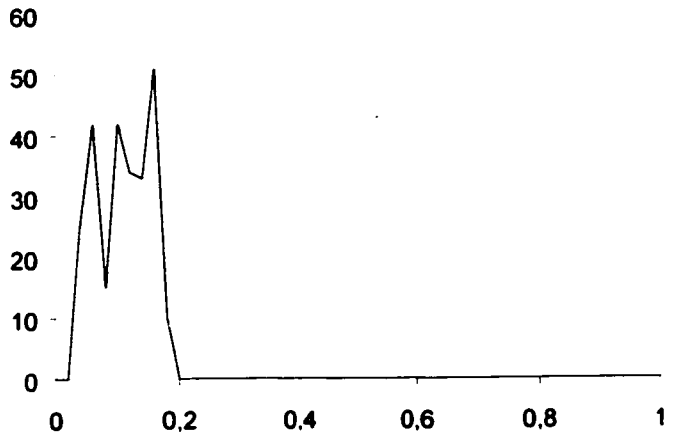


Figure 3.4: Histogram of the quality factor of the triangles in Figure 3.3. All of them are below 0.2

3.2.3 Recovering the geometry in an STL mesh

The contents of an STL file consists of the vertex coordinates of each triangle and the associated normal. In order to achieve a complete meshing of the surface, we need to

have some data on the topology and the curvatures of the surface to be meshed. The topology and connectivity between triangles is obtained by avoiding redundancy between vertices in the STL file. This is done by using a binary tree in which the vertices are stored and sorted according to a lexicographic order. At the end of this procedure, we have a mesh of the object's surface, with all the types of connectivity needed for the re-meshing, particularly the connectivity used for adjacency searches.

Then, as many industrial artefacts have edges, we need to determinate where they are located, if any. This is done using an edge detection process, considering the connectivity between entities in the triangulation.

The search for the model's edges is very important since they have to be kept intact in the re-meshing, i.e. they must not be cut by the Delaunay re-meshing strategy. The search is based on angle calculations between adjacent triangles. The common segment of two triangles is considered to be an edge if the angle between the normal vectors of the two triangles is greater than a specific value (typically 20°). It should be mentioned that, in addition, we can handle specific edges, such as the contact area between two different parts in an assembly (in order to apply boundary conditions, for example). These segments will be considered as edges, and will be kept in the final mesh.

Figure 3.5: Recovering the edges (black lines)

The curvature is then evaluated at each element of the triangulation by seeking the other triangles connected to it and by examining their normal vectors (see 6). The following procedures are used: For every triangle T_i, we iterate on its three vertices V_j. $(j = 0...2)$. At each vertex, there are two sides, S_{jk} $(k = 0...1)$, and the associated triangles adjacent to T_i, T_{jk}. If S_{jk} is a sharp edge, we define a parameter r_{jk} as 0; otherwise it is 1. The curvature C_{T_i} is evaluated as follows:

$$C_{T_i} = \max_{j=0\ldots2}\left(\frac{|\theta'_j|}{\frac{\|S_{j0}\|+\|S_{j1}\|}{2}\cdot|\sin(\theta_j)|}\cdot\prod_k r_{jk}\right) \tag{22}$$

Where θ_j is the angle between S_{j0} and S_{j1}, θ'_j is the angle between the normal vectors of T_{j0} and T_{j1}. After this, the curvature is extrapolated at each vertex V_i (C_{V_i}), using the maximum curvature for the n triangles connected to V_i:

$$C_{V_i} = \max_{j=0..n-1}(C_{T_j}) \tag{23}$$

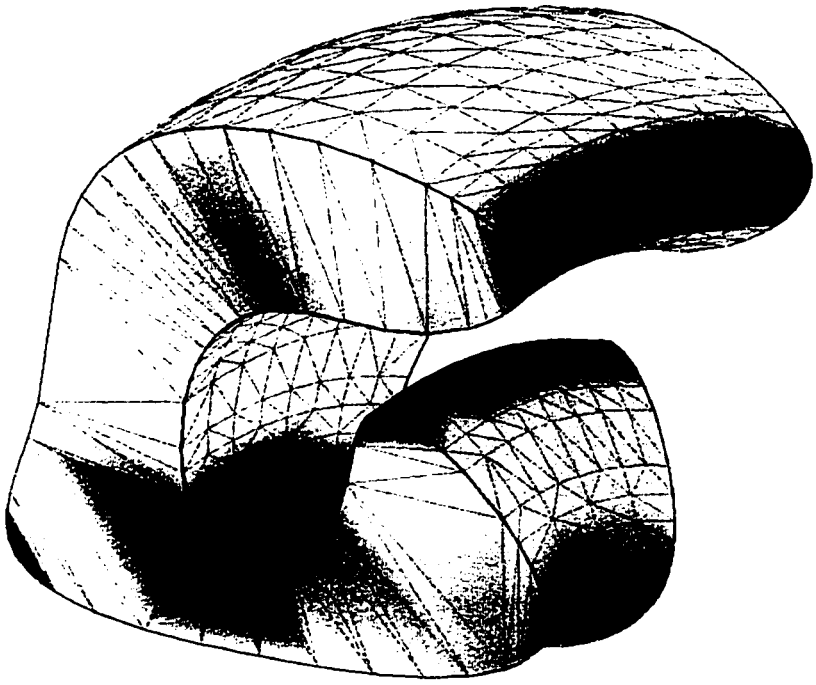


Figure 3.6: Graph of the curvatures obtained on the triangles by linear interpolation of the curvatures corresponding to the vertices. The most curved areas are shown in black

We should mention that the curvature considered here is a norm of the tensor of curvatures related to the surface (i.e. we do not care about the directions along which the effects of the curvature take place), and it should be noted that the curvatures obtained are not very accurate. This is because of the discrete geometrical representation of the surface (set of planes).

In fact, these curvatures will only drive an optional adaptive mesh refinement. The result is a coarser mesh in less curved areas, and refined elsewhere. Figure 3.6 illustrates the mesh resulting from a linear interpolation of the curvature across each triangle using the equations mentioned above.

At this stage, the information we need to begin the mesh generation procedure is the information about the isotropic size map of the target mesh. Since this is not the aim of the present research work we assume either that a constant size map is provided by the user, or that a size map based on an error estimator has been provided (from a previous FEM calculation). We also consider the case where the density map is based on a geometric criterion (controlling the maximum distance between the STL mesh and the triangulation).

3.3 Refinement method

The triangles of the mesh are sorted with respect to a reduced size ratio (ratio between the actual size and the target size). The worst triangles (the biggest ones) are the first to be handled using the bisection algorithm described in the sequel. At each stage, the sorting is updated (since there are new triangles and some have been destroyed) and the process continues as long as all the triangles do not meet the criterion. When all the triangles match the criterion, we apply a smoothing procedure. The resulting mesh is ready to deal with the FEM.

3.3.1 Bisection algorithm

The new vertices are generated by bisection of a line segment that already exists in the mesh. The vertices are generated successively on segments in the mesh. They are not necessarily part of the surface, so the vertices must be projected onto the initial STL mesh.

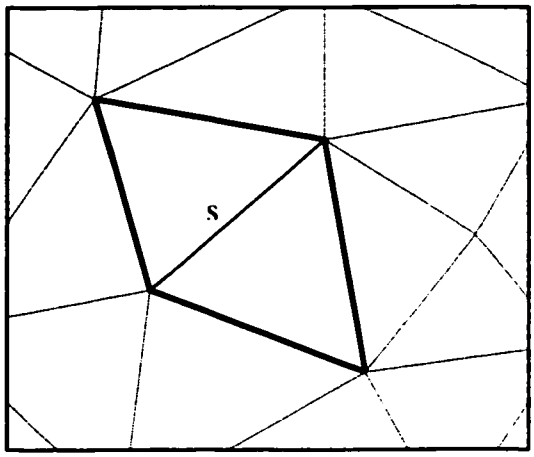


Figure 3.7: Before bisection of the segment S

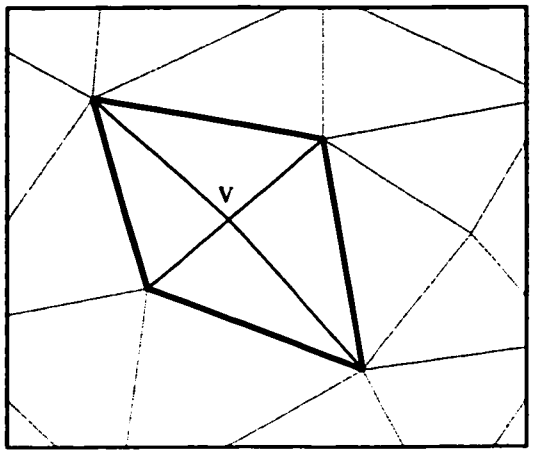


Figure 3.8: After bisection. The new vertex V has been created

The line segment on which we will generate the vertex is found using the algorithm that follows, which is taken from [9]: First, we search for the triangle T to be refined, the one that least respects the target size, as described above. Then we search for a path composed of triangles that have a side longer than T. This defines a sequence (T_i).

Let S_n be the longest segment of the triangle T_n. The triangle T_{n+1} is adjacent to the triangle T_n along its longest segment S_n. The sequence stops provided that S_n and S_{n+1} are the same segments. The sequence begins, of course, with T_0=T. The first segment to be bisected is the last segment in the sequence (S_i). The algorithm is repeated until the first segment we considered, S_0, is bisected. This leads to a correct refinement of the area considered. The sequence described here is valid if one uses a triangle-based refinement procedure. If instead, segments are classified according to their size and will then be refined, it is useless to construct such a sequence as the longest edge is already known.

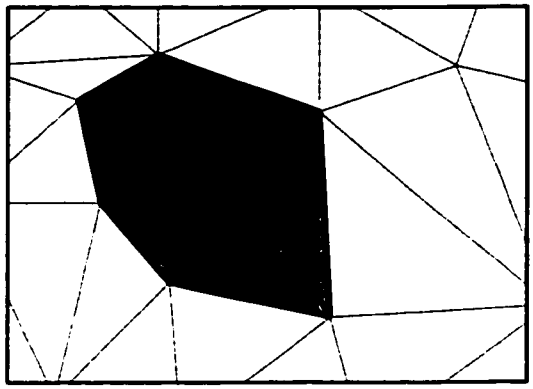


Figure 3.9: Sequences of triangles and segments, before bisection of the segment S_n

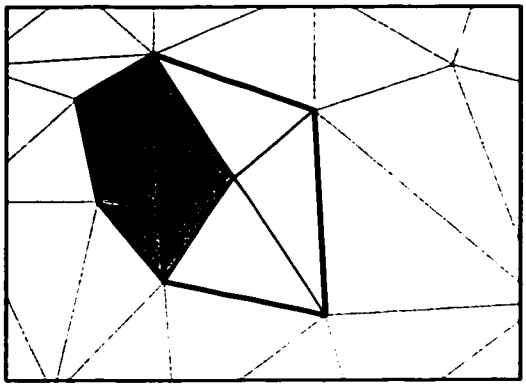


Figure 3.10: Bisection of the last segment of the sequence, S_n

3.3.2 Respecting the Delaunay criterion

After each bisection, the mesh must be handled locally in order to achieve better regularity. This is done by respecting the Delaunay criterion, which is well-known for 2D meshes. However, we are dealing with curved surfaces and therefore the conventional Delaunay criterion must be changed. This is in fact essential in order to deal with the curvature of the surface and, of course, its discrete representation. Let us return to the original definition of a Delaunay triangulation, in two dimensions [4,5,10]:

> **Def. 1**: Property of the empty circle: Let T be an arbitrary triangulation of the convex hull of a set of vertices S. If the property of the empty circle is verified for every configuration of two adjacent triangles of the triangulation T, then this means that it is verified for the whole triangulation. The triangulation T is referred to as a Delaunay-conforming triangulation.

This definition can be adapted to curved surfaces, taking into consideration their curvatures. It is important to note that if we change the definition of the metric, the Delaunay criterion is still valid [2,6]. We should consider a new definition of the

circumcircle. In this case the circumcircle is the location of all points on the surface whose curvilinear distance to the centre is the same, namely the radius. The curvilinear distance is calculated using an integral. In the parametric space (if it can be locally considered as constant), the image of the circumcircle as defined just above is generally an ellipsis. Unfortunately, as the surface is only known through the STL triangulation, such a parametrization is not well defined.

We propose using another criterion, which is of course less accurate than the former criterion, but is still good enough for testing topologically near triangles. This forms the basis for definitions 2 and 3, which are taken from [8].

> **Def. 2:** Property of the empty sphere: Let T_s be an arbitrary triangulation of the convex hull of a set of vertices S located on a surface. If the property of the empty circumscribed sphere is verified for every configuration of two adjacent triangles of the triangulation T_s, then T_s is referred to as a weak Delaunay-conforming triangulation.
>
> **Def 3:** If, in addition, the property of the empty sphere is verified for the whole triangulation, then T_s is referred to as a strong Delaunay-conforming triangulation on the surface.
>
> **Def 4:** The circumscribing sphere to a triangle is the sphere with the same centre and the same diameter as the circumscribing circle.

It should be noted that triangulations that are weakly Delaunay-conforming in the sense of Definition 2 generally do not respect the empty sphere criterion for the whole surface, i.e. they are not strongly Delaunay-conforming in the sense of Definition 3. This is not an obstacle when dealing only with surface meshing, but if subsequently we have to generate a volume mesh using a 3D Delaunay method, then some triangles will eventually have to be cut [8]. If we are dealing with planar geometry, Definition 2 comes

to Definition 1 with no more conditions, and we can affirm that in this case the triangulation is Delaunay-conforming provided that it is weakly Delaunay-conforming, as in Definition 2.

In Definition 2, the distances are measured in the 3-dimensional Euclidean space. Thus, instead of testing a node with an exact Delaunay criterion, we verify only if it is not in the circumscribing sphere of the triangles located in the vicinity of the node. This vicinity is related to the topology of the mesh. It is, in fact, the connectivity relationship between the triangles that defines the vicinity (the shell concept, [6]).

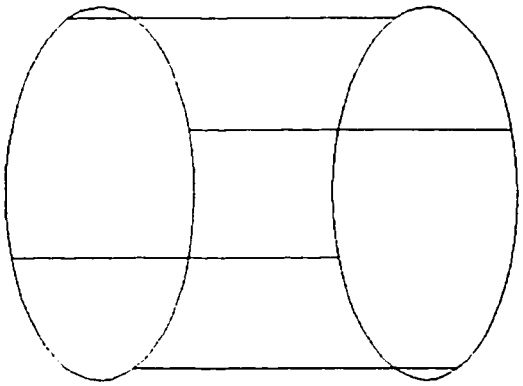


Figure 3.11: Cylinder, designed using Autodesk AutoCad

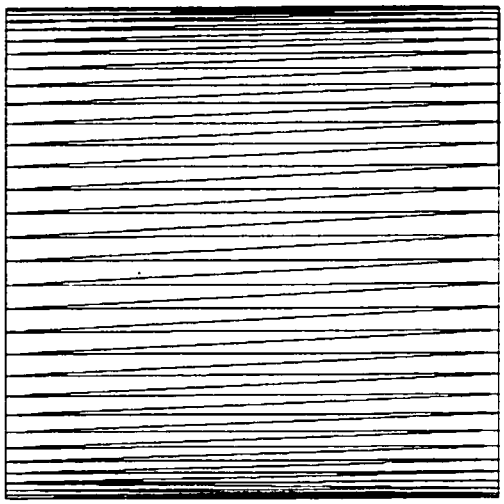


Figure 3.12: STL mesh of the surface generated by AutoCad (side view)

The local remeshing algorithm is based on diagonal swapping [1]. The diagonals to be swapped are determined by searching for triangles that do not respect the modified Delaunay criterion relative to the new vertex. We perform a kind of "star" remeshing of the convex domain that is composed of the triangles found previously, by connecting the added node to the corners of the convex domain. This is the "Watson" algorithm [11]. In order to avoid poor geometrical approximation, the search for triangles that do not

respect the modified Delaunay criterion is done only on triangles whose orientation does not overly differ from the orientation of the triangles in the immediate vicinity of the new vertex, a difference of about 15° to 60° being normal, see Figure 3.13 and Figure 3.14 (same cylinder than in Figure 3.12). To avoid destruction of the edges of the object, in the algorithm it is forbidden to swap segments that belong to the list of edges, as defined above.

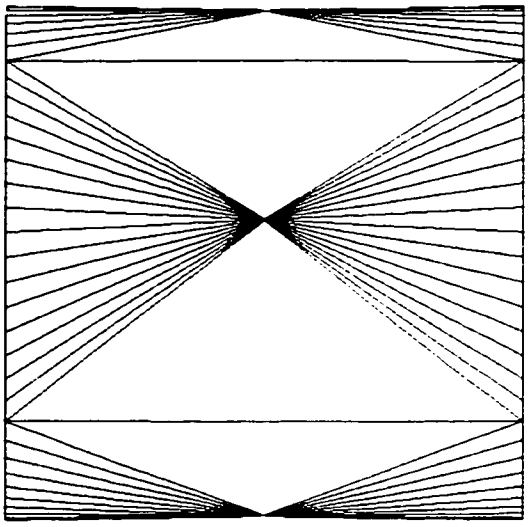


Figure 3.13: Several meshing steps, with a threshold angle of 45° (4 vertices inserted)

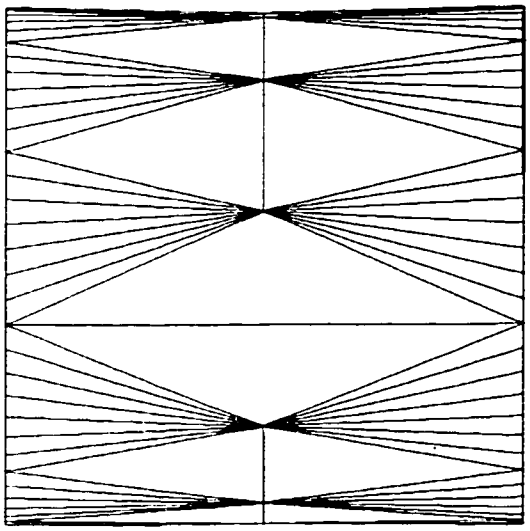


Figure 3.14: Several meshing steps, with a threshold angle of 20° (8 vertices inserted)

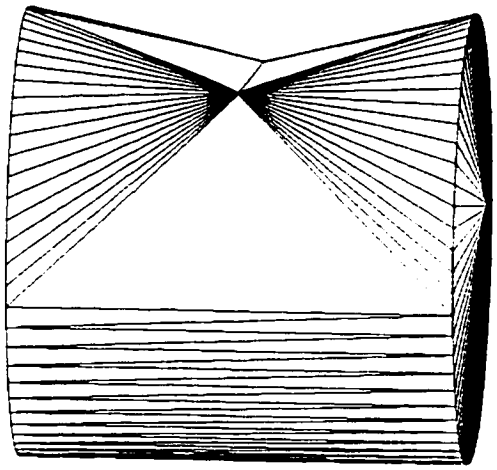


Figure 3.15: Poor geometrical approximation of the surface: threshold angle of 45°

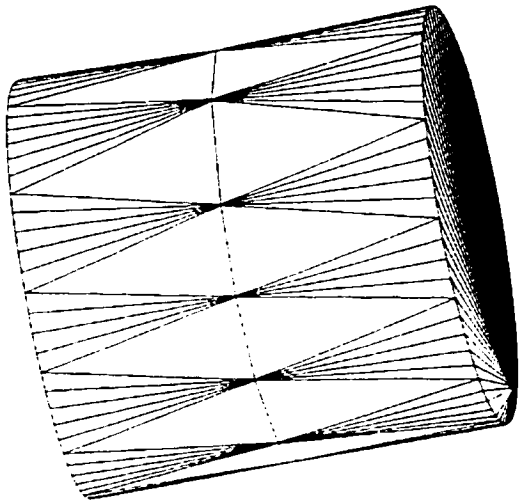


Figure 3.16: Good approximation: threshold angle of 20°

Figure 3.13 to Figure 3.16 show the effect of the threshold angle on the quality of the surface approximation during the initial meshing steps. As anticipated, the edges are not destroyed.

The fact that not all the triangles that do not respect the modified Delaunay criterion are taken into account is not a problem because the original STL mesh will be highly refined. Indeed, from a certain point of advancement of the algorithm, triangles that do not respect the modified Delaunay criterion will all be taken into account because at that point the difference in orientation between all the triangles will be below the angle threshold. At the end of the refining step, the triangulation will be weakly Delaunay-conforming. This is the key in adapting the algorithm described in [9] to curved surfaces.

3.3.3 The projection algorithm

As we see in Figure 3.15, Delaunay remeshing leads to poor geometric approximations. Often, when we generate a node on a segment that is not part of the original STL mesh, it is not located on the original STL model. This is because of the remeshing that occurs around every inserted node (Watson algorithm). Consequently, we need a projection algorithm to relocate the node onto the surface as described by the original STL triangulation.

This operation is time-consuming because the target triangle on the initial STL mesh on which the vertex will be projected is not known a priori. An adjacency search algorithm is used to avoid testing all the triangles of the initial STL mesh. This algorithm is also useful when the solid to be meshed features very thin volumes (see example 3.4: interleaved tetrahedrons). We should avoid the projection of a node belonging to the inner surface onto the outer surface. The projection algorithm is based on the fact that each new node is created on a segment (due to the bisection algorithm). In fact, we know the STL triangle where a new node is created. Using information in the vicinity between the STL triangles, it is easy to test a limited number of triangles. In order to do so, we start by testing an STL triangle that corresponds to one of the nodes of the segment that is bisected. We then test all neighbouring triangles and proceed until we find the proper triangle onto which the new triangle will be projected. Simple geometric algebra is used for the actual projection. A similar approach is used for mesh smoothing. When a node is moved during the smoothing process, its new location should be on the STL triangulation. This, of course, implies that the information about the initial STL triangulation has to be kept throughout the whole process.

3.3.4 Smoothing

We implemented a kind of "laplacian" smoothing, obtained by moving the vertices at the location of the barycentre of the connected vertices, without relaxation. This operation is repeated for each "free" vertex of the mesh. Of course, we have to re-project the vertices onto the STL surface. Again, we are not concerned with vertices that are situated on edges (they are not free to move). It should be noted that the laplacian smoothing of a surface without limits or edges (such as a sphere) can't converge, because the mesh is to move indefinitely. This is a poorly formulated problem, somewhat similar to a laplacian problem without boundary conditions. However, our aim is to obtain a local smoothing so that we do not have to wait until complete convergence is achieved. In most cases, ten iterations are sufficient. By local smoothing, we mean that the regularity of the mesh is related to how the nodes are set relative to each other; clearly, the influence of nodes farther away is much less than nodes in the immediate vicinity, so we don't need to wait until complete convergence is achieved to obtain a regular mesh.

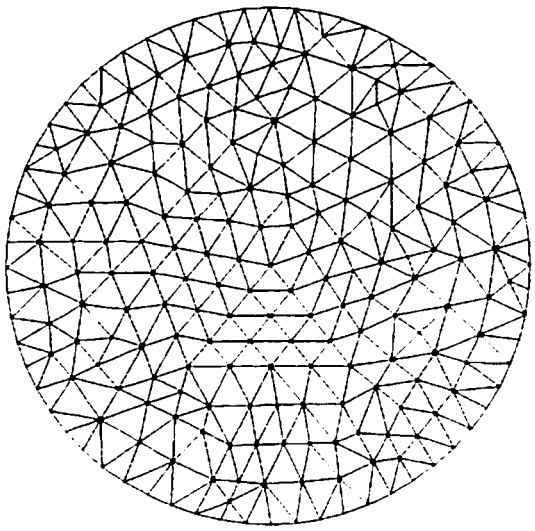


Figure 3.17: Before smoothing

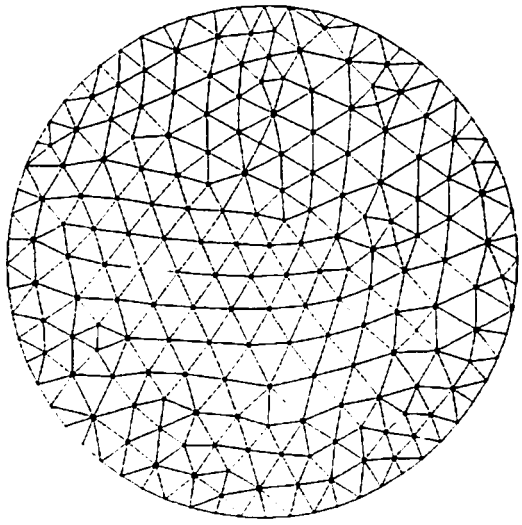


Figure 3.18: After smoothing (10 iterations)

3.3.5 Respecting a size map

As shown in the examples below, algorithms have been implemented in order to respect different types of size maps that are imposed a priori. Since nodes are generated on existing triangles using bisection techniques (as long as the length of the sides of the triangle are locally greater than the prescribed size), the actual mesh size is about 0.6 times the prescribed size.

The size map consists of either a constant size map, an arbitrary size map provided by the user or by an error estimator, or a size map that will lead to a good geometric approximation of the solid's boundary. In the latter case, mesh density is driven locally by the maximum geometric error (theoretically the gap between the target mesh and the boundary of the original solid model). As the original solid model is not available (we only start from the STL triangulation), we use the curvatures defined above in order to approximate the solid boundary locally and calculate the gap. The gap considered here,

for a given triangle T_i, is the maximum distance between T_i and the sphere that passes through the vertices of the triangle, with R_{T_i} being the radius of the sphere (see Figure 3.19):

$$R_{T_i} = \frac{1}{C_{T_i}} \tag{24}$$

where C_{T_i} is the curvature in the vicinity of triangle T_i.

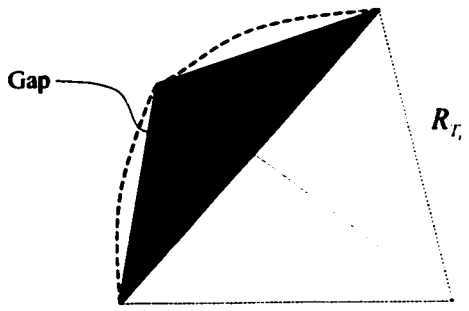


Figure 3.19: Definition of the gap between the sphere and the triangle

3.3.6 User input

The only parameters that have to be provided by the user of our mesh generation process are the following:

- the STL file
- the target size or the size map
- the threshold angle for the edge detection algorithm, represented by T_e. In the examples shown below T_e has been set to 45°.

- the threshold angle for avoiding poor geometric approximation during the Delaunay remeshing process, represented by T_d. In the examples shown below T_d has been set to 25°. It should be mentioned that T_e and T_d are closely related. If $T_d < T_e$, the edges detected with T_e will not be destroyed anyway.

3.3.7 The advantages of using STL instead of CAD patches

In a mesh generation process based on CAD patches, the patches that make up the solid's boundary are meshed individually. Consequently, the edges shared by adjacent patches are kept in the resulting mesh. Most of the time, patches in CAD designs are not related to any physical meaning and generally there is no point in keeping patch edges in the resulting mesh for FEM calculation purposes. Moreover, keeping patch edges often leads to a stretched triangle because they contribute to constraining the resulting mesh. In our mesh generation system the only edges that are kept are "true" edges and edges that are needed to apply the boundary conditions. The following example (see Figure 3.20) illustrates the concept of "patch independent" mesh generation.

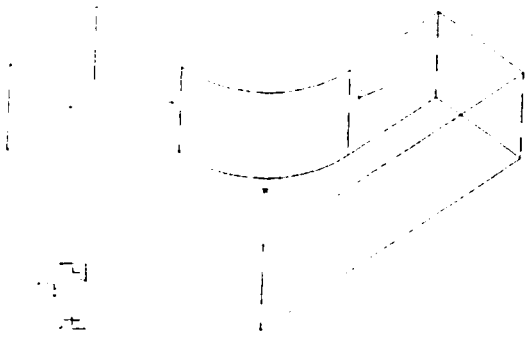


Figure 3.20: CAD model of a fillet

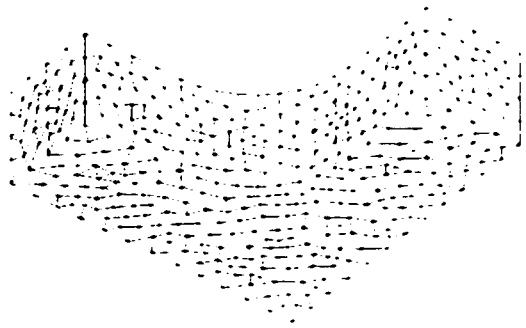


Figure 3.21: Resulting mesh obtained from the STL fille (patch independent)

3.4 Validation examples

3.4.1 Cylinder

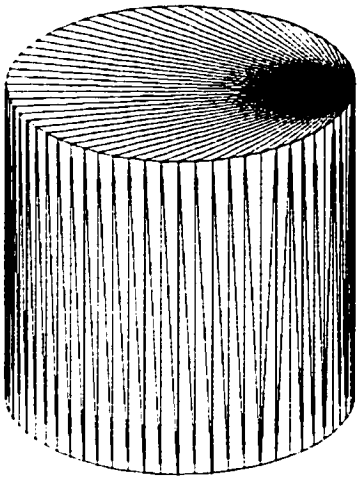


Figure 3.22: Original STL mesh. 252 triangles. 128 vertices. Height: 30 mm

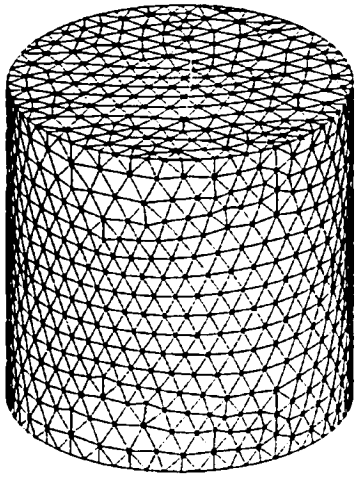


Figure 3.23: Refinement: constant target size: 3mm. Smoothing. 2302 triangles. 1153 vertices

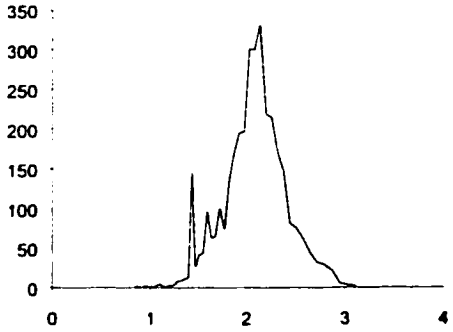


Figure 3.24: Histogram of the length of segments corresponding to Figure 3.23. The target size is 3 mm

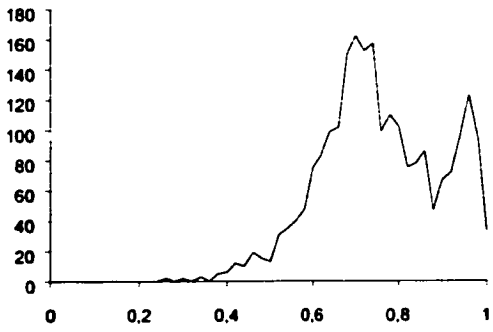


Figure 3.25: Histogram of the quality factor of the triangles in Figure 3.23

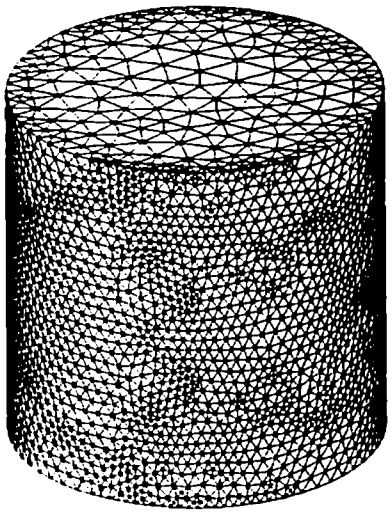


Figure 3.26: Adaptive refinement: maximum imposed geometric error: 0.005mm. maximum imposed triangle size: 4 mm for low curvature regions. Smoothing. 7086 triangles, 3545 vertices.

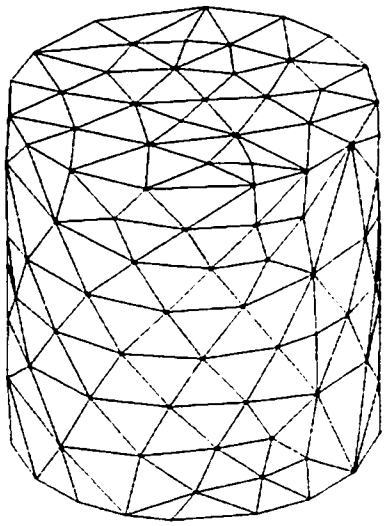


Figure 3.27: Refinement: constant target size: 7mm, coarsening (not described here): constant target size: 5 mm. Smoothing. 314 triangles. 159 vertices.

3.4.2 Revolution solid

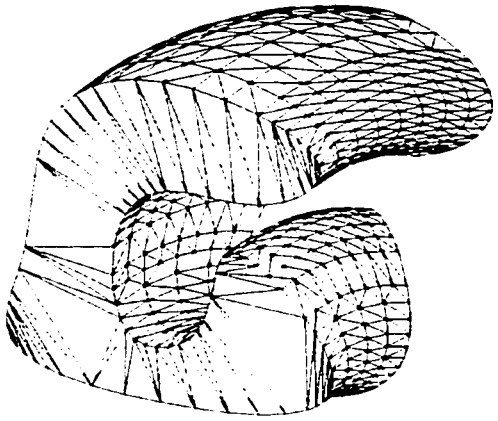


Figure 3.28: Original STL mesh. 1386 triangles. 695 vertices. Size: 150 mm

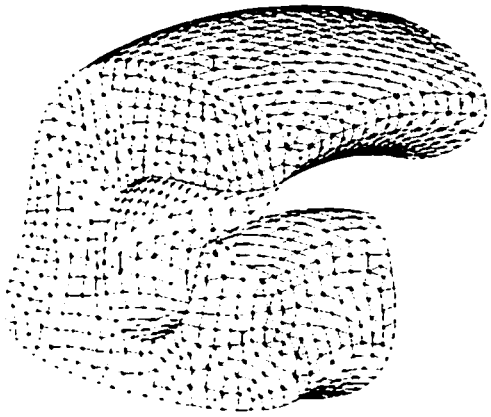


Figure 3.29: Refinement: constant target size: 8 mm. Smoothing. 4040 triangles, 2022 vertices

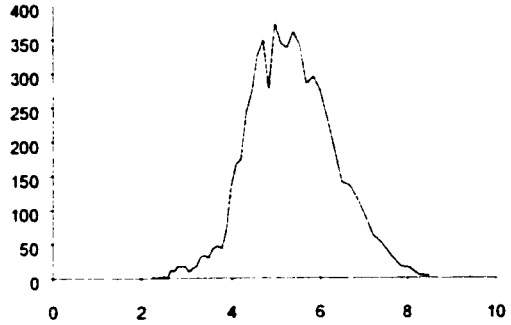


Figure 3.30: Histogram of the length of segments corresponding to Figure 3.29. The target size is 8 mm

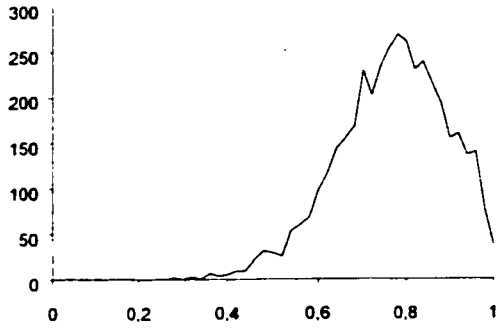


Figure 3.31: Histogram of the quality factor of the triangles in Figure 3.29

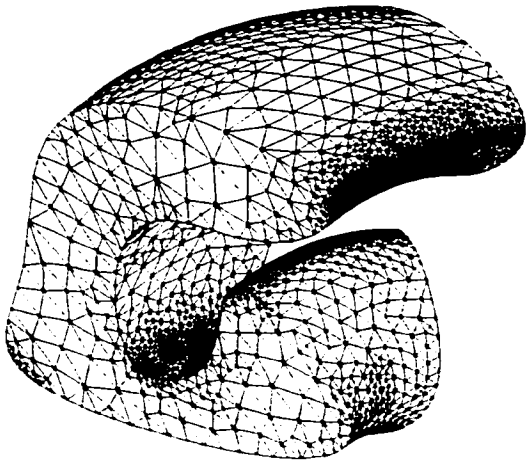


Figure 3.32: Adaptive refinement: maximum imposed geometric error: 0.1 mm, maximum imposed triangle size: 20 mm for low curvature regions. Smoothing. 6434 triangles. 3219 vertices.

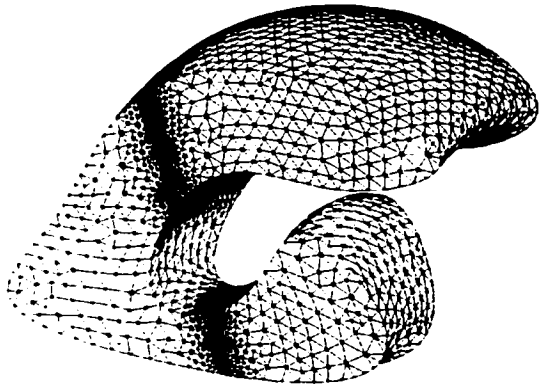


Figure 3.33: Refinement: constant target size: 8 mm , except on a plane: 2 mm. Smoothing. 8622 triangles, 4313 vertices.

3.4.3 Power supply fan

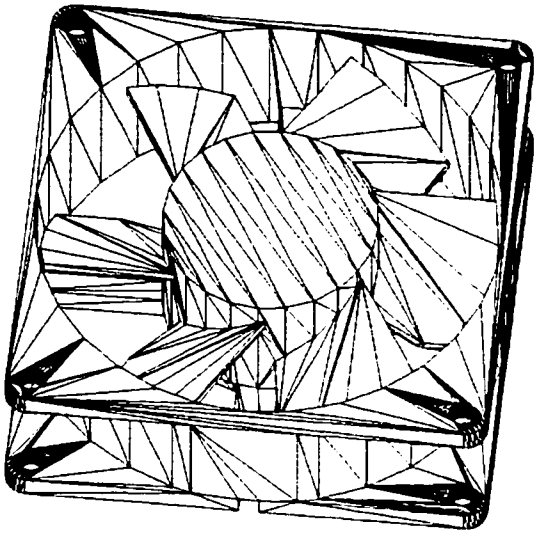


Figure 3.34: Original STL mesh. 1908 triangles, 932 vertices. Size: 100 mm

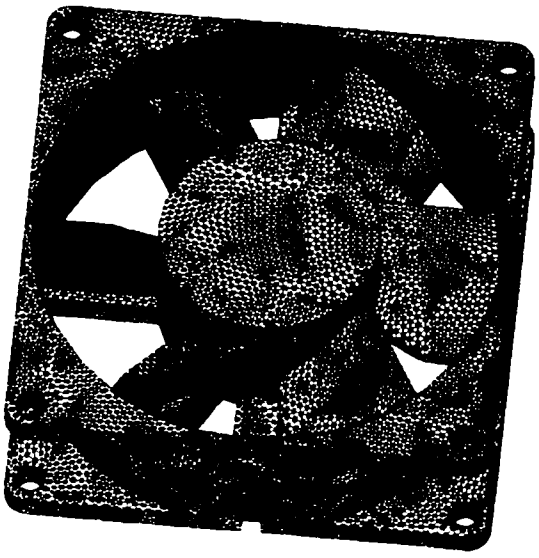


Figure 3.35: Refinement: constant target size: 2mm. Smoothing. 59196 triangles, 29576 vertices

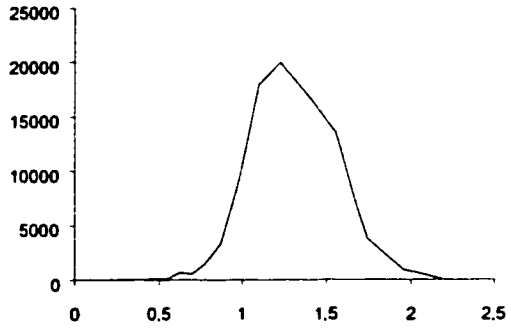


Figure 3.36: Histogram of the length of segments corresponding to Figure 3.35. The target size is 2 mm

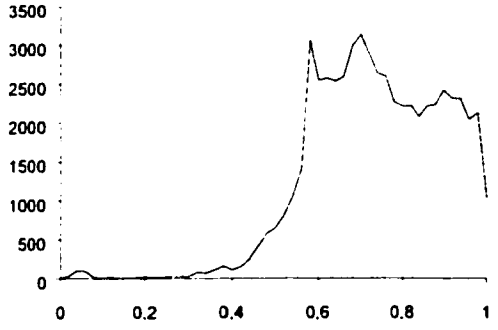


Figure 3.37: Histogram of the quality factor of the triangles in Figure 3.35. Thin blades imply quality factors under 0.1

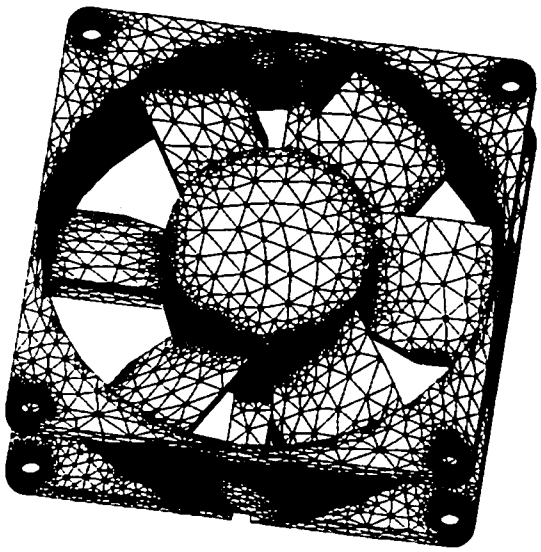


Figure 3.38: Adaptive refinement: maximum imposed geometric error: 0.01 mm. maximum imposed triangle size: 10 mm for low curvature regions. Smoothing. 75696 triangles, 37826 vertices.

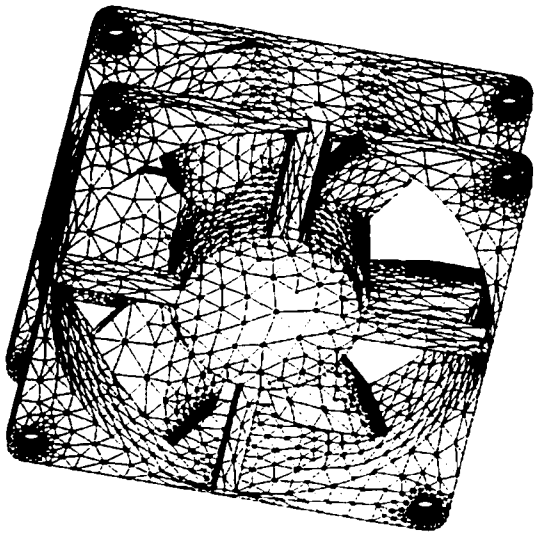


Figure 3.39: Adaptive refinement: maximum imposed geometric error: 0.1 mm. maximum imposed triangle size: 10 mm for low curvature regions. Smoothing. 13730 triangles. 6843 vertices

3.4.4 Interleaved tetrahedrons

The difficulty comes from the fact that there are 5 tetrahedrons touching each other (distinct solids). The projection algorithm could project vertices onto the wrong solid, and this is avoided by adjacency relations used in the projection algorithm described above.

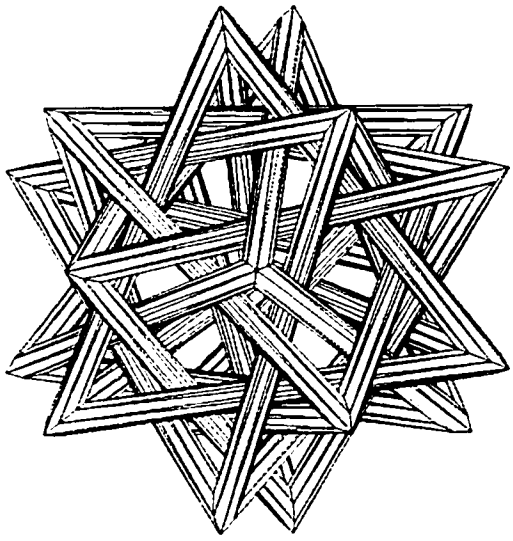


Figure 3.40: Original STL mesh. 480 triangles, 220 vertices. Size: 60 mm

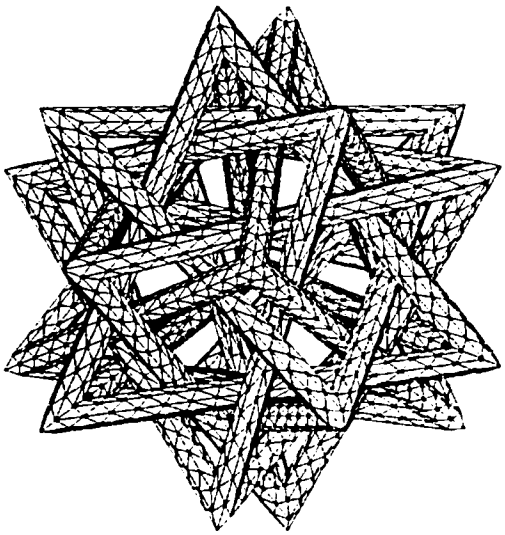


Figure 3.41: Refinement: constant target size: 2mm. No smoothing. 7920 triangles, 3940 vertices

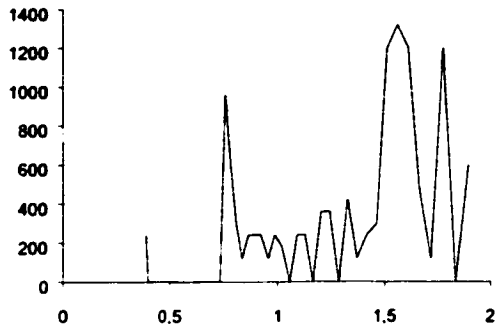


Figure 3.42: Histogram of the length of segments corresponding to Figure 3.41. In this case, the histogram is not properly smoothed. This is due to the fact that the geometry heavily constrains the shape of triangles and the length of segments. The target size is 2 mm.

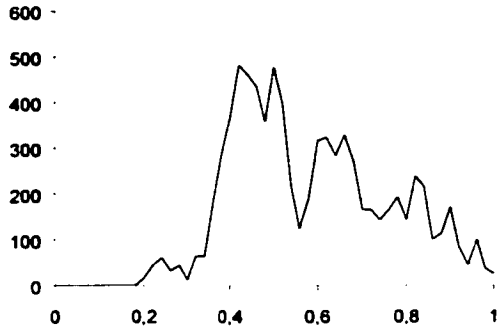


Figure 3.43: Histogram of the quality factor for triangles in Figure 3.41. Same remark as for Figure 3.42.

3.4.5 Guitar

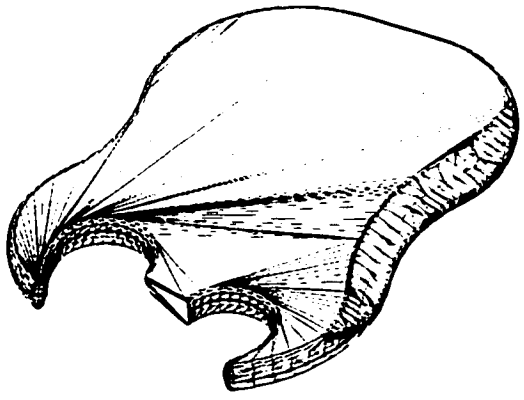


Figure 3.44: Original STL mesh. 3470 triangles, 1737 vertices. Size: 20 inches

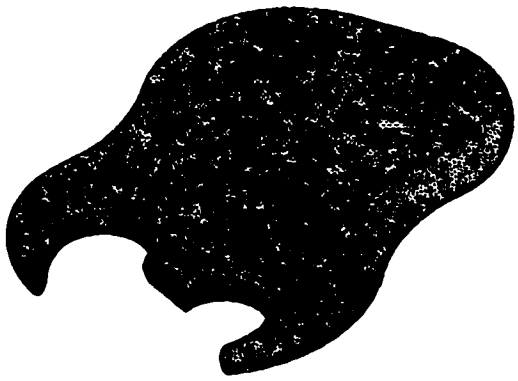


Figure 3.45: Refinement: constant target size: 0.2 in. Smoothing. 45602 triangles, 22803 vertices

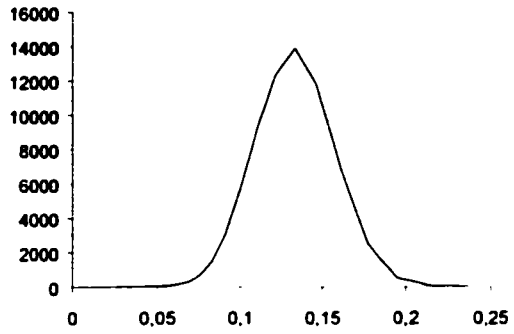


Figure 3.46: Histogram of the length of segments corresponding to Figure 3.45. The target size is 0.2 inches

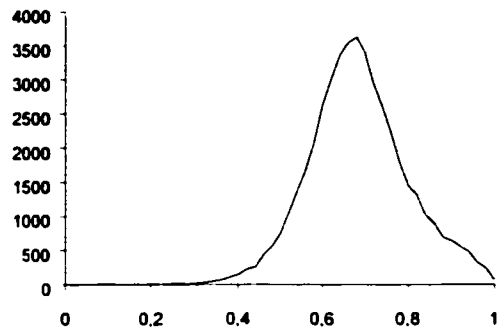


Figure 3.47: Histogram of the quality factor for the triangles in Figure 3.45

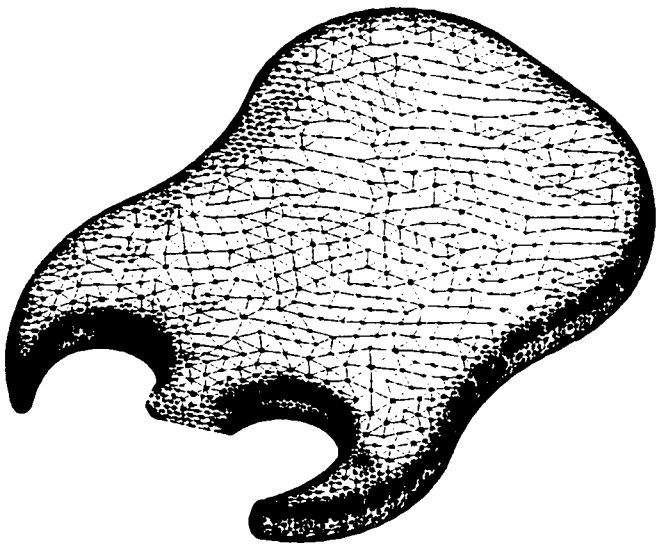


Figure 3.48: Adaptive refinement: maximum imposed geometric error: 0.02 in., maximum imposed triangle size: 0.8 in. For low curvature regions. Smoothing. 27550 triangles, 13777 vertices

3.4.6 Lever

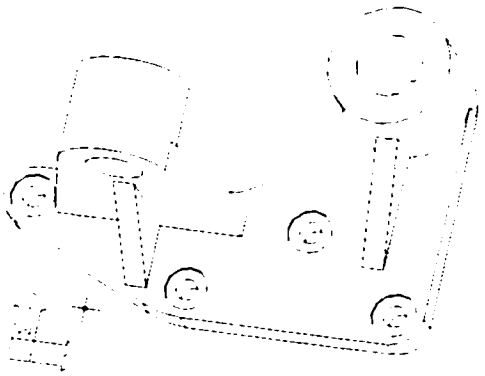


Figure 3.49: AutoCAD model of a lever

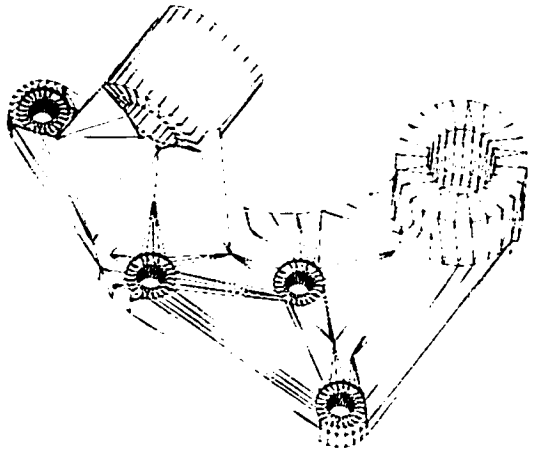


Figure 3.50: Original STL mesh. 1330 triangles, 655 vertices. Size: 377 mm

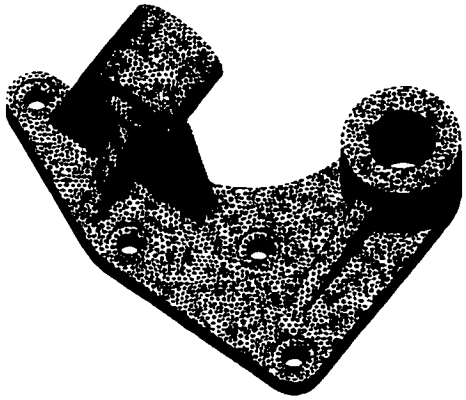


Figure 3.51: Refinement: constant target size: 5mm. Smoothing. 30208 triangles, 15094 vertices

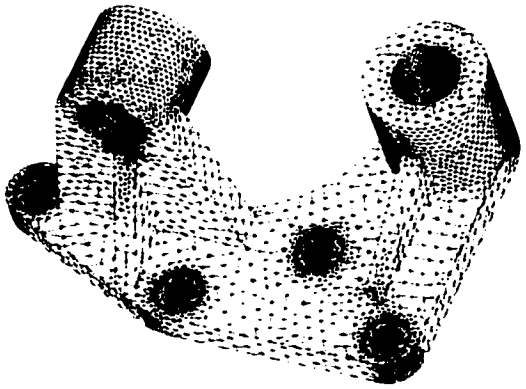


Figure 3.52: Adaptive refinement: maximum imposed geometric error: 0.05 mm, maximum imposed triangle size: 13 mm for low curvature regions. Smoothing. 26432 triangles, 13206 vertices

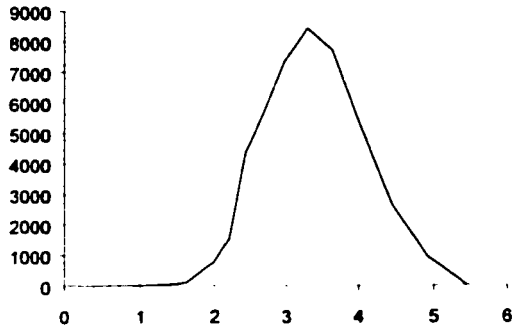


Figure 3.53: Histogram of the length of segments corresponding to Figure 3.51. The target size is 5 mm.

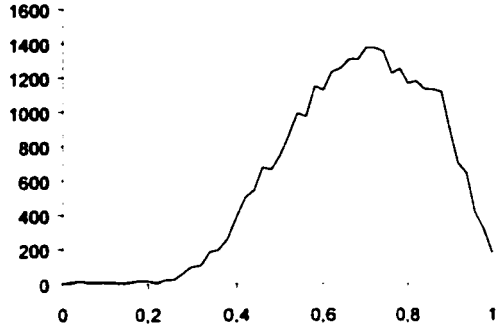


Figure 3.54: Histogram of the quality factor for the triangles in Figure 3.51.

3.4.7 Aneurysm

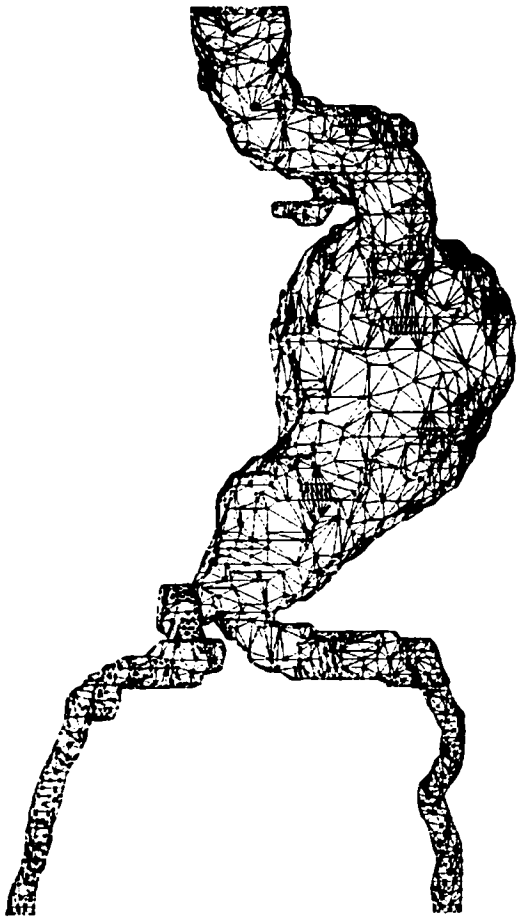


Figure 3.55: STL mesh from a medical scanner. 10494 triangles, 5245 vertices. Size: 278 1/10mm Courtesy of Dr. M.L. Raghavan, Department of Biomedical Engineering, University of Iowa, Iowa City, IA

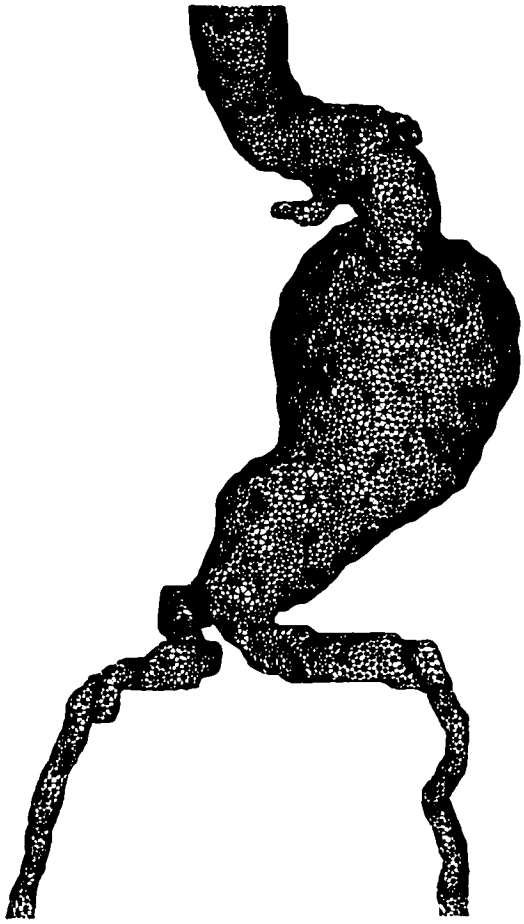


Figure 3.56: Refinement : constant target size : 3 1/10mm). Smoothing. 21462 triangles, 10729 vertices

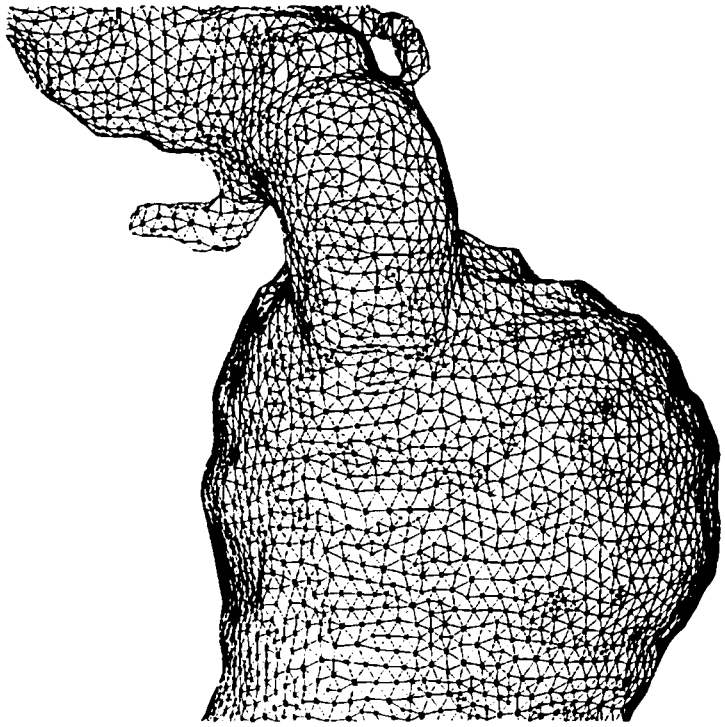


Figure 3.57: Closeup of Figure 3.56.

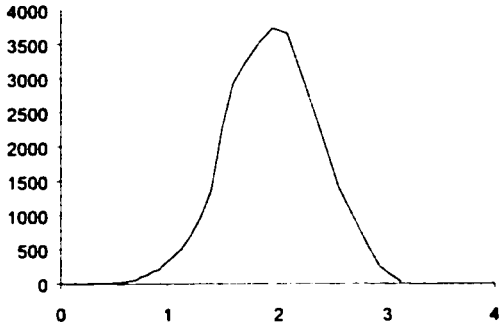


Figure 3.58: Histogram of the length of segments corresponding to Figure 3.56. The target size is 3 mm

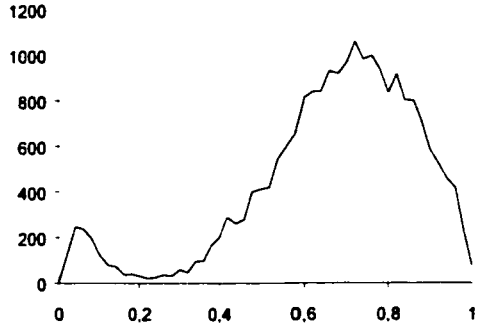


Figure 3.59: Histogram of the quality factor for the triangles in Figure 3.56. Because of tiny details on the original model, there are still some stretched triangles

3.5 Conclusions

In this paper, we have shown the possibility of directly re-meshing (i.e. without knowing the actual exact geometry) not only meshes from STL files, but more generally all surface meshes, provided that they are conforming. For example, this can be done when using a FEM adaptive remeshing loop and a size map provided by an error estimator.

The method described can mesh a solid without being restricted by the CAD representation (patches), i.e. two patches can be meshed without their common boundary being in the resulting mesh. The result is a patch-independent mesh, which can be advantageous for very complex parts composed of many small patches. In addition, the mesh procedure is independent of any exact CAD file format. We currently mesh geometry that originates with many CAD systems, such as AutoCAD and Catia.

Further work on anisotropic mesh generation using the same basis will be carried out.

3.6 References

[1] C. Cherfils, F. Hermeline, 1990, Diagonal swap procedures and characterizations of 2D-Delaunay triangulations, Math. Mod. And Num. Anal. **24** (5), 613-626.

[2] J.C. Cuillière, 1997, A direct method for the automatic discretization of 3D parametric curves, Comp. Aided Design **29** (9), 639-647.

[3] J.C. Cuillière, 1998, An adaptive method for the automatic triangulation of 3D parametric surfaces, Comp. Aided Design **30** (2), 139-149.

[4] B. Delaunay, 1934, Sur la sphère vide, Bul. Acad. Sci. URSS, Class. Sci.Nat., 793-800.

[5] G.L. Dirichlet, 1850, Über die Reduktion der positiven quadratischen Formen mit drei understimmten ganzen Zahlen. Z. Angew. Math. Mech. **40**(3). 209-227

[6] P.L. George, H. Borouchaki, 1997, Triangulation de Delaunay et maillages : application aux éléments finis, Paris, Hermès. ISBN 2-86601-625-4

[7] Y. Ito, K. Nakahashi, 2000, Direct surface triangulation using stereolithography (STL) data. 38th AIAA Aerospace Sciences Meeting & Exhibit. AIAA-2000-0924.

[8] P.P. Pébay, P.J. Frey, 1998, A priori Delaunay-conformity. Proceedings of the 7th International Meshing Roundtable. 321-333.

[9] M.C. Rivara, P. Inostroza, 1997. Using longest-side bisection techniques for the automatic refinement of delaunay triangulations. Intl. J. for Num. Meth in Eng. **40**. 581-597.

[10] G. Voronoï, 1908, Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Recherches sur les parallélоèdres primitifs. Journal Reine angew. math. **134**.

[11] D.F. Watson, 1981, Computing the n-dimensional Delaunay Tesselation with application to Voronoï polytopes. Computer Journal **24** (2). 167-172.

CHAPITRE 4
RE-MESHING ALGORITHMS APPLIED TO MOULD FILLING SIMULATIONS IN RESIN TRANSFER MOULDING

E. BÉCHET[1,2], E. RUIZ[1], F. TROCHU[1,*], J.-C. CUILLIERE[2]

(1) Centre de Recherches Appliquées Sur les Polymères (CRASP), Département de Génie Mécanique, École Polytechnique de Montréal, H3C 3A7, Canada

(2) Laboratoire de productique, Département de Génie Mécanique, Université du Québec à Trois-Rivières, G9A 5H7, Canada

E-mail: eric.bechet@epost.de

4.1 Abstract

In injection moulding processes such as *Resin Transfer Moulding* (RTM) for example, numerical simulations are usually performed with a fixed mesh, on which the displacement of the flow front is predicted by the numerical algorithm. During the injection, special physical phenomena occur on the front, such as capillary effects inside the fibre tows or heat transfer when the fluid is injected at a different temperature than the mould. In order to approximate these phenomena accurately, it is always better to adapt the mesh to the shape of the flow front. This can be achieved by implementing re-meshing algorithms, which will provide not only more accurate solutions, but also faster calculations. In order to represent precisely the shape of the saturated domain in the cavity, the mesh needs to be non isotropic in the vicinity of the flow front. The size of the elements along the front is connected to the overall accuracy needed for the simulation (as interpolation error); also it is also connected with the curvature of the front (one need more elements along the front if it is very curved). The size in the perpendicular direction governs the accuracy on the position of the moving boundary in time. Since these two constraints on element size are not related, the need for non isotropic mesh refinement is crucial if one want to optimise simulation time versus

simulation error. In the approach proposed here, the mesh is changed at each time step from a coarse background mesh used as starting point in the refinement algorithm. The solution needs to be projected on the new mesh after each re-meshing. This amounts to adopting a new filling algorithm, which will be validated by comparison to a standard simulation (without re-meshing) and with experimental data.

Keywords: Resin transfer moulding, simulation, finite elements, anisotropic mesh generation, level sets.

4.2 Introduction

The numerical schemes used in mould filling simulations are usually based on a time dependent resolution of an unsteady (free surface) boundary value problem. Because the boundary of the filled area in the mould cavity is constantly evolving, it is difficult to generate a fixed mesh suitable for all the successive calculation steps of a filling simulation. This leads to numerical difficulties in capturing some physical phenomena that occur on the flow front, such as in the RTM process the heat exchanged between heated fibres and the resin. Other related problems concern the calculation time and computer memory needed to perform accurate flow predictions for complex geometries. In existing standard numerical procedures, a fine mesh is required everywhere in the mould cavity. This means that the solution time grows at an exponential rate with the desired accuracy, usually in the following way:

$$t \approx k(d) \cdot s^{d+1} \tag{25}$$

where s is the density of the mesh (in arbitrary units), and d stands for the topological dimension of the domain (2 or 3), provided that the semi-bandwidth of the linear system remains stable when the mesh density increases. Parameter $k(d)$ is a constant independent of s.

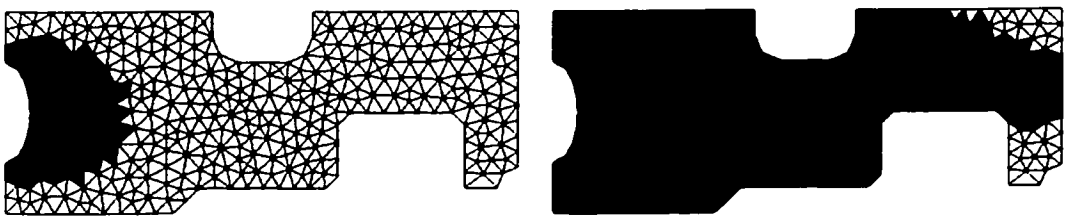


Figure 4.1: Filled elements (in grey) in a mould filling simulation on a fixed mesh

As shown in Figure 4.1, the front position cannot be approximated with a fine precision on a fixed mesh. However, this simulation performed with LCMFlot software [26] presents the advantage of conserving the volume of resin in the mould, because the numerical scheme is based on linear non conforming finite element approximation with degrees of freedom defined at the middle of the edges. These have the mathematical property of conserving the resin flow rate across inter-element boundaries (for a discussion about this choice, see [27]). The time stepping mechanism considered here is simple as there is no dependence between time steps (the problem here is quasi-static at each time step, so there is no need of a specific time integration scheme). In order to avoid a poor representation of the flow front, it is necessary to refine the mesh at each calculation step. In such an adaptive scheme, there is no need for the tangential node density (d_t) along the front to be the same as the normal density (d_n). In fact, d_t is related only to the spatial accuracy of the resolution at each time step, whereas d_n has an influence on the accuracy of the position in time of the flow front during the whole simulation. As a consequence, a non isotropic mesh generation algorithm is required here. Moreover, the use of an adaptive refinement scheme will lower dramatically the need of computational resources for a given numerical accuracy.

Chang and Kikuchi [14] have used a re-meshing procedure in their simulation of the RTM process. This work was focused on the numerical solution of Darcy's equation at each time step in the saturated domain, but no assumption was made on the accuracy of the flow front. The objective of this paper is to develop a non isotropic re-meshing algorithm that will follow the moving boundary and allow to predict its evolution. This article is divided into three main parts: (1) non isotropic mesh generation; (2) the creation of a suitable metric field to control the mesh size; and (3) the update of the front flow after each time step. Finally, the re-meshing algorithm is interfaced with the mould filling simulation software LCMFlot to study the performance and accuracy of the numerical solution. The re-meshing algorithm is compared with an experiment and with the results of the commercial software LCMFlot (cf. `www.esi-group.com`).

4.3 Surface mesh generation

An initial mesh must be provided, that will be fine enough to represent the geometry of the part, typically a STL file most of the time. A standard bisection algorithm will be used to refine the original mesh [13].

4.3.1 Isotropic scheme

New vertices are generated by bisection of a line segment already existing in the mesh. A new vertex is created near the middle of the longest line segment (hence increasing the density of the mesh). This vertex must be inserted topologically and projected on the initial mesh (see Figure 4.2).

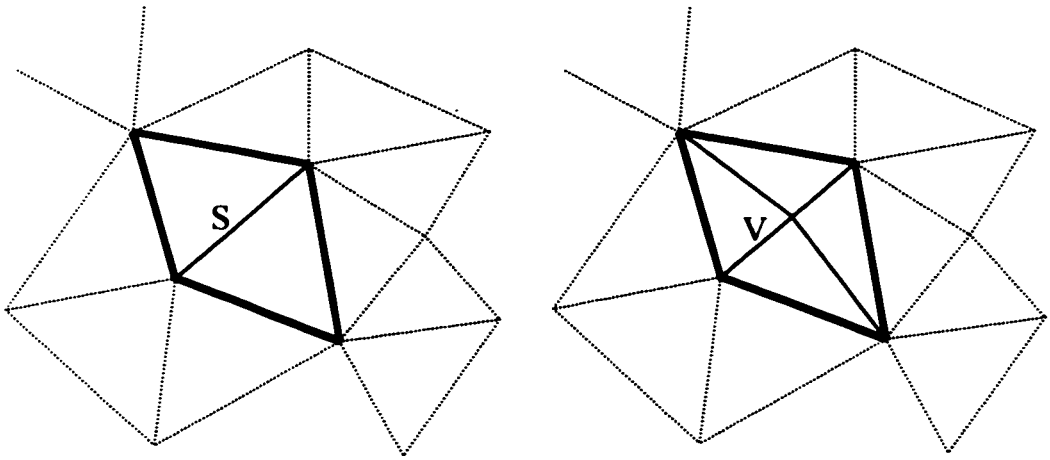


Figure 4.2: Insertion of a new vertex 'V' on an existing edge 'S'

After each bisection, the mesh must be modified locally in order to achieve a better regularity in the size of the elements. This is done by respecting the well-known Delaunay criterion for 2D meshes [23]. However, since in the general case we deal with curved surfaces, the conventional Delaunay criterion must be adapted here to the

discrete representation of a curved surface with the initial background mesh. Let us return to the original definition of Delaunay triangulation in two dimensions [18][19][29].

> **Def. 1** - Property of the empty circle: Let T be an arbitrary triangulation of the convex hull of a set of vertices S. The property of the empty circle means that the circumscribed circle to a triangle of the mesh does not contain any other node than those of that triangle. If this property is verified for every configuration of two adjacent triangles, then it is verified for the whole triangulation T which will be referred to as a Delaunay-conforming triangulation.

This definition leads to the most regular mesh obtainable from a given set of nodes (the mesh which minimizes the maximum angle in all triangles). This criterion can be adapted to curved surfaces by taking their curvature into account. However, a parametric representation of the surface is needed, which is not available in the general case. Therefore a modified criterion is proposed, which is of course less accurate than the former one, but that will allow to test to test topologically nearby triangles. This leads to definitions 2 and 3, taken from [22].

Def. 2 - The circumscribing sphere to a triangle is the sphere with the same centre and the same diameter as its circumscribing circle.

Def. 3 - Property of the empty sphere: Let Ts be an arbitrary triangulation of the convex hull of a set of vertices S located on a surface. The property of the empty circumscribed sphere means that the circumscribed sphere to a triangle does not contain any other node than those belonging to that triangle. If it is verified for every configuration of two adjacent triangles, then the triangulation Ts is referred to as a weak Delaunay-conforming triangulation.

Def. 4 - If, in addition, the property of the empty sphere is verified for the whole triangulation, then Ts is referred to as a strong Delaunay-conforming triangulation of the surface.

Note that triangulations that are weakly Delaunay-conforming in the sense of Definition 2 generally do not respect the empty sphere criterion for the whole surface, i.e., they are not strongly Delaunay-conforming. However, this does not represent a problem here, as no 3D mesh needs to be generated. The local re-meshing algorithm is based on Watson's algorithm, which consists of a local improvement of the mesh by edge swapping without having to consider the whole mesh [30]. In order to avoid a poor geometrical approximation and a destruction of the real edges of the geometric object, the algorithm forbids to swap edges belonging to sharp angles as well as to boundaries of the surface. The insertion of new vertices stops when the size of the longest segment becomes of the desired size. Figure 4.3 shows an example of surface mesh generation using this method.

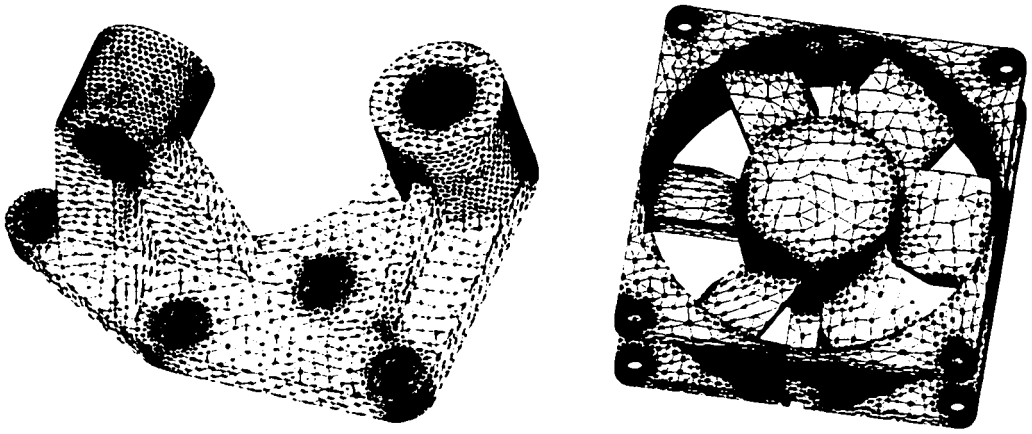


Figure 4.3: Examples of isotropic surface triangular meshes

4.3.2 Non isotropic extension

As we need here to deal with anisotropy, the refinement algorithm must be adapted to generate a mesh related to a non isotropic size map. A non isotropic size map indicates how stretched triangles should appear everywhere on the surface. It is defined by the desired size in orthogonal directions and by the orientations of these directions. In two-dimensions, three independent parameters are required to define a non isotropic size map. This can be summarized with a 2x2 symmetric positive definite matrix, called the *metric*, or a similar metric field (function of position) if the element size is non constant over the computational domain. The notion of distance between two points is then redefined as follows [20]:

$$dist(AB) = l(\Gamma) = \int_0^1 \sqrt{{}^t\mathbf{s}'(t) \cdot \mathbf{M}(s(t)) \cdot \mathbf{s}'(t)}\, dt \qquad (26)$$

where A and B are the points considered, $\mathbf{s}(t)$ is a parametric representation of the path followed on the surface to go from A to B (usually a straight segment), and $\mathbf{M}$ is the metric field. The desired size at every position depends on the metric field. In fact,

when evaluated with the proposed metric, the length of any line segment of the mesh is set close to one by the refinement algorithm. This allows to choose the longest segment (with respect to the size map in the metric field) to be refined first.

In our work, this distance is approximated by Simpson's integration scheme. However, if the metric is locally constant along the line segment AB, the following quadrature formula can be used:

$$dist(AB) = \frac{\sqrt{{}^t\overrightarrow{AB} \cdot \mathbf{M}(A) \cdot \overrightarrow{AB}} + \sqrt{{}^t\overrightarrow{AB} \cdot \mathbf{M}(B) \cdot \overrightarrow{AB}}}{2} \tag{27}$$

Of course, the latter expression is faster to compute, but less accurate if there are great variations in the metric. The above formula cannot be used at the beginning of the meshing algorithm because line segments of the background mesh cross all the domain (and the moving boundary). Note that the numerical scheme may degenerate if these distances are not calculated with enough accuracy.

Finally, definitions 1 to 4 of Delaunay triangulations must be changed. The structure remains the same, but, of course, circles are changed to ellipses, and spheres to ellipsoids. Figure 4.4 shows two non isotropic meshes generated by two arbitrary analytical size maps. Both provide a constant isotropic mesh, except in the vicinity of a sinusoidal path. The first size map is stretched normally above the sine wave, and stretched tangentially under. The second one is only stretched tangentially.

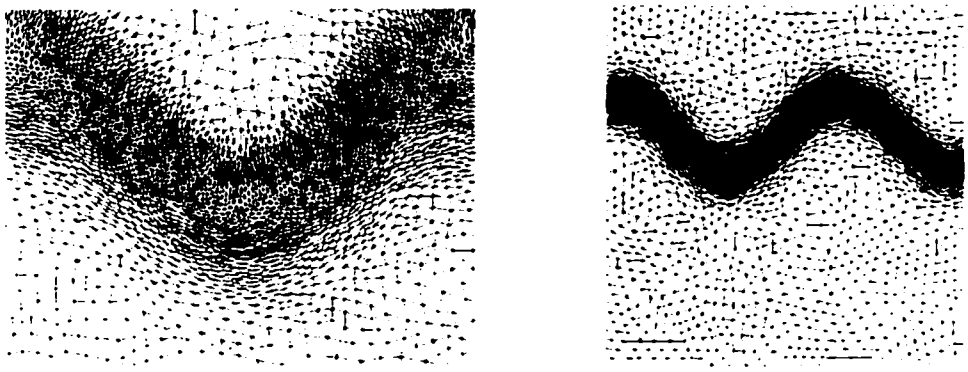


Figure 4.4: Examples of 2D non isotropic meshes

4.3.3 Determination of a Non isotropic size map

The mesh generation algorithm needs a size map in order to create a suitable mesh. This size map is generated from physical parameters (current front positions, geometry of the mould) and computational parameters (required accuracy, efficient usage of computational resources, calculation speed).

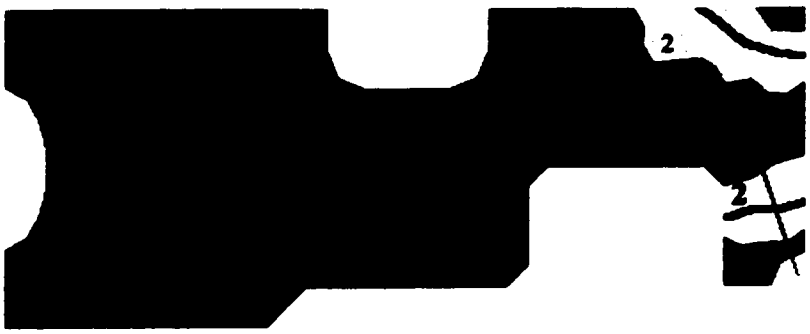


Figure 4.5: Zones in the mould cavity (thick black lines denote the positions of the flow front)

As shown in Figure 4.5, there are 3 different zones to consider:

(1) the *fully saturated zone*, where the solution of Darcy's boundary value problem is computed (isotropic mesh);
(2) the *front neighbourhood*, where high thermal gradients may exist (non isotropic mesh);
(3) the *empty zone*, where almost nothing happens, apart from heat exchanged by conduction (isotropic mesh).

A filling factor F between 0 and 1 is usually associated to each element of the mesh. It represents the ratio of resin volume contained in an element to the pore volume. This factor can segregate easily between regions (1) and (3). However, a constant-by-element filling factor is not precise enough to represent a continuous front. Therefore a continuous nodal representation will be adopted here in order to model the position of the front. In order to define the front neighbourhood (2), a scalar field is needed that will represent the distance to the front. The position of the moving boundary is determined by taking the 0.5 iso-value of the filling factor (0 represents an empty pore, and 1 a fully saturated volume). The distance of each node of the (previous) mesh to the front will be calculated by the fast marching method described later.

The next question is related to the motion of the flow front from one position to the other at each calculation step. For this purpose, an efficient filling algorithm is needed to update the position of the resin front. It is not computationally efficient to recalculate the pressure and velocity fields after each new layer of elements is filled up with resin. An improved algorithm to predict the position of the resin front at each calculation step is needed here. For that purpose, a time field representing the arrival time of the resin at each node of the mesh must be calculated. This time field requires the knowledge of extrapolated velocities in the *empty zone* and of a distance field representing for every node the "distance" to the front.

4.3.4 Representation of the filling factor

The filling factor F associated to the nodes of a finite element mesh is a discontinuous variable (i.e., $F = 1$ if a node lies in the filled region; $F = 0$ if it is in an empty region). Standard finite element approximations cannot represent such a field if the discontinuity does not match with element boundaries. This is the case in a mould filling problem, because during the simulation there is no *a priori* information on where the flow front will move from its current position. Instead, a continuous representation designed to keep track of the exact position of the front shall be used here.

For the sake of clarity, this will be illustrated in Figure 4.6 for a one-dimensional example. Elements are numbered from left to right (E1) to (E6), and the nodes of the "mesh" are denoted from (a) to (f). The discontinuity is located at A between nodes (c) and (d). The figure shows the "theoretical" discontinuous filling factor above, which cannot be represented directly on a finite element mesh. The continuous approximation is plotted below as a piecewise linear continuous function in each element. By convention, the 0.5 iso-value will be taken as the exact position of the moving front in time. Note, however, that this kind of approximation does not allow an exact conservation of the fluid mass as it is. In the next equation, let F be the "true" filling factor (discontinuous), and F^* the continuous approximation of F. With these notations, the filled volume V_r can be defined as :

$$V_r = \int_\Omega F(x) \cdot dx = \int_{\Omega_f} 1 \cdot dx \neq \int_\Omega F^*(x) \cdot dx \qquad (28)$$

where Ω is the whole volume of the mould, and Ω_f is the volume such that $F^*(x) > 0.5$. The volume of fluid calculated with the continuous nodal representation F^*

of the filling factor F is not necessarily equal to the exact fluid volume V_r. This approach is also valid in the two- and three-dimensional cases.

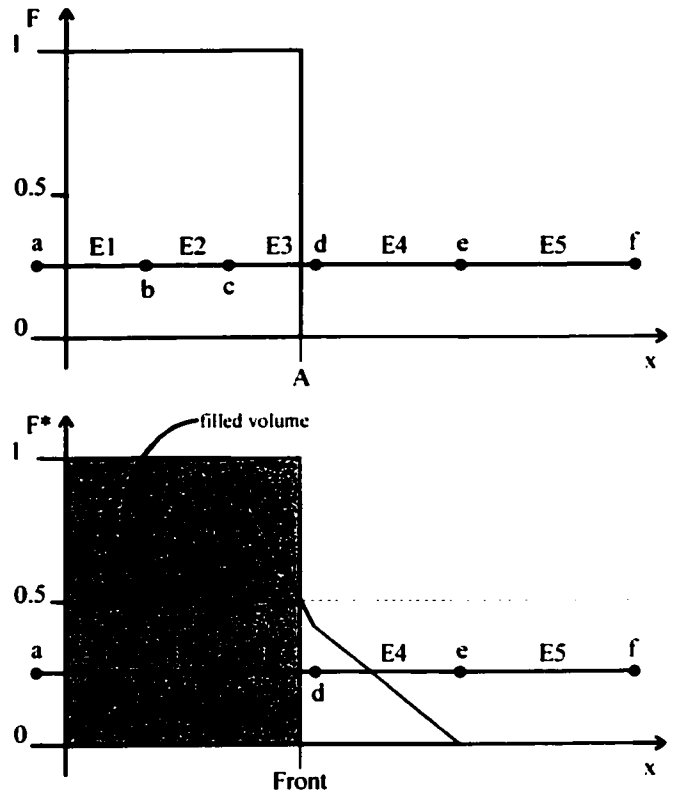


Figure 4.6: Filling factor

4.3.5 Notion of level set

Sethian et al. [12][24][25] developed a mathematical tool to describe time-evolving phenomena, including moving boundary problems. Let $v(\mathbf{X},t)$ denote the normal velocity to the front at a position $\mathbf{X}$ and time t. Basically, a level set is a function $u(\mathbf{X},t)$, which is solution of the following "initial value" problem :

$$\frac{\partial u}{\partial t} + v(\mathbf{X},t)\cdot|\nabla u| = 0 \text{ in } \Omega \tag{29}$$

$$u(\mathbf{X},t) = 0 \text{ on } \Gamma(t) \tag{30}$$

$$\Gamma(t_0) = \Gamma_0 \tag{31}$$

where $\Gamma(t)$ denotes the moving boundary at time t, $\Gamma(t_0)$ being the initial position of the front. Equation (29) is derived from the transport equation of a scalar field $u(\mathbf{X},t)$:

$$\frac{\partial u}{\partial t} + \mathbf{V}(\mathbf{X},t)\cdot\nabla u = 0 \text{ in } \Omega \tag{32}$$

where $\mathbf{V}(\mathbf{X},t)$ is the velocity vector of the fluid. If $\mathbf{n}$ denotes the normal vector at time t and position $\mathbf{X}$ on the interface, we have the following identities :

$$v(\mathbf{X},t)\cdot|\nabla u| = \mathbf{V}(\mathbf{X},t)\cdot\mathbf{n}\cdot|\nabla u| = \frac{\mathbf{V}(\mathbf{X},t)\cdot\nabla u\cdot}{|\nabla u|}|\nabla u| = \mathbf{V}(\mathbf{X},t)\cdot\nabla u \tag{33}$$

The velocity vector $\mathbf{V}(\mathbf{X},t)$ is supposedly defined everywhere in Ω. In moving boundary problems, it is not necessary to know the tangential component of the velocity to the interface; only the normal component $v(\mathbf{X},t)$ plays a role. The function $u(\mathbf{X},t)$ provides a mathematical mean to follow the moving position of the interface in time. It is an arbitrary function of position and time, with some constraints. A distance function to the front at a given time (see corresponding paragraph) will meet the requirements. These requirements are that this function must be continuous so that its gradient will be defined at the interface and it must vanish along the front (interface). In fact, $u(\mathbf{X},t)$

plays a similar role for an interface as the characteristic function of a set. $\Gamma(t)$ represents the shape of the front (interface) at instant t. It needs to be known at an initial time t_0, as well as the normal velocity function $v(\mathbf{X},t)$ in the whole spatial and temporal domain. These equations are valid for any v, both positive and negative. Every point $\mathbf{X}$ where $u(\mathbf{X},t)=0$ lies on the boundary at time t, thus allowing to track its successive positions by taking the iso-value $u=0$ at related time steps $\{t_0,t_1,t_2\ldots,t_n\}$.

4.3.6 Eikonal equation

In the case of a monotonous evolution of the moving boundary (which is true for the resin front in RTM), this investigation can be restricted to the more specific case of the so-called Eikonal equation :

$$v(\mathbf{X})\cdot|\nabla u|=1 \text{ in } \Omega \tag{34}$$

$$u=g(\mathbf{X}) \text{ on } \Gamma \tag{35}$$

These equations define a "boundary value" problem at time t_i, the solution of which is $u(\mathbf{X})$ and has here a physical meaning (unlike the solution $u(\mathbf{X},t)$ of the level-set equations). It represents the time when a point $\mathbf{X}$ will be reached by the moving boundary, if the displacement of the front is extrapolated from its current position. This is also called the relative "arrival time" at position $\mathbf{X}$. The reference time for the current position of the front is set to zero. Clearly, $u(\mathbf{X})$ must be single-valued, thus the restriction to positive values for $v(\mathbf{X})$. Physically, this means that the front cannot revert itself once it has reached a certain position. Once the assumption on positive

values for $v(\mathbf{X})$ is made, the evolution of the moving boundary becomes monotonic, thus permitting a time independent solution scheme.

When dealing with a flow in a porous medium (Darcy's law), there are no dynamic effects need to be considered, so the sign of the normal pressure gradient remains constant along the front. Thus, the front cannot move backwards. This is sufficient to prove that the latter mathematical background of the Eikonal equation is appropriate, with the normal velocity function to the front $v(\mathbf{X})$ being always positive.

We are now going to solve the Eikonal equation to evaluate a "distance function" of every node to the front.

4.3.7 Distance function

Solving the Eikonal equation for a unitary normal speed $v(\mathbf{X}) = 1$ and an homogeneous boundary condition $u = 0$ on Γ leads to a solution $u = d(\mathbf{X})$ which can be considered as a *distance function* from the point $\mathbf{X}$ to the curve Γ, namely the current position of the flow front. In a discrete computational domain, this function will be extremely useful to locate the nodes of the mesh with respect to the moving boundary. Note that this distance is not Euclidean. This will be illustrated easily in the next section, which describes how the distance of any node to the front is calculated.

4.3.8 Computation of the distance field

The solution of the Eikonal equation in a triangulated discrete domain is described in detail in [25]. This is an hyperbolic problem, in which the "information" is transmitted from the boundary conditions to the whole domain. Hence, a solution scheme based on

the displacement of a moving front can be used here, much similar to the fast marching method described in [21].

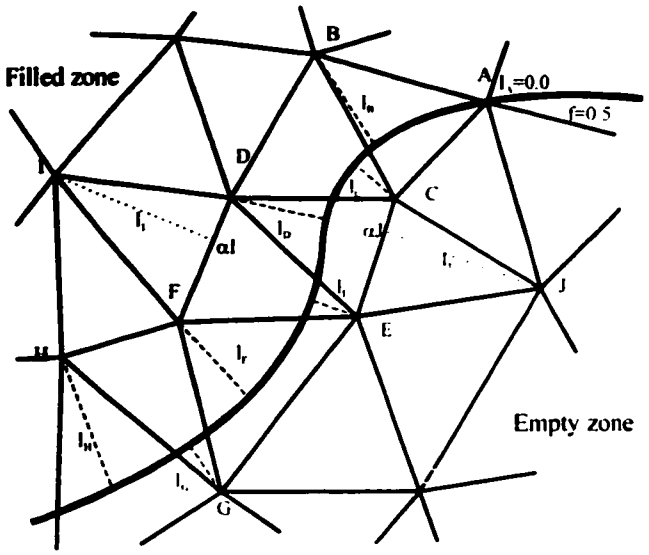


Figure 4.7: Distance evaluation scheme

The geometrical approach developed here is illustrated in Figure 4.7, which shows in bold the position of the moving boundary at a given time. The mesh displayed here and in similar graphics in the sequel is isotropic for readability purpose only. Nodes situated on the front have a distance value equal to zero (for example at node "*A*", $d_A = l_A = 0$). In the first layer of elements, the distance associated to each node is the length of the segment perpendicular to the iso-value (nodes concerned in Figure 4.7 are *B*, *C*, *D*, *E*, *F*, *G*, *H*). For example, $d_C = l_C$ at node "*C*". For any subsequent layer, the algorithm proceeds layer by layer until all nodes have been processed. Hence the distance d_K for any node "*K*" is evaluated by the following equation:

$$d_K = \left(\alpha_K \cdot d_{N_{1a}} + (1-\alpha_K)\cdot d_{N_{1b}}\right) + l_K \qquad (36)$$

where d_{N_0} and d_{N_1} are the distances associated to the nodes of the previous layer and l_K denotes the Euclidian length of the projection of node "K" on the segment N_0N_1. Parameter α_K is the relative position of this projection on the segment N_0N_1 (i.e., if the projection lies in the first quarter, $\alpha_K = 0.25$). Naturally, for each layer of elements, nodes are sorted relatively to their distance. The zone affected by these computations grows in the same way as distance iso-values do. If a node is encountered with a distance already known (for example, if layers collide), the new distance will be computed anyway. The shortest value will be chosen as new distance for this node. This ensures the construction of a consistent and continuous distance field, which can be considered as a piecewise linear interpolation of the "real" distance inside each triangle. Figure 4.8 shows the distance field determined from a given flow front position on an isotropic mesh. The distance field allows to locate each node in the mesh with respect to the flow front. Based on this approach, a non isotropic size map adapted to the flow can now be constructed.

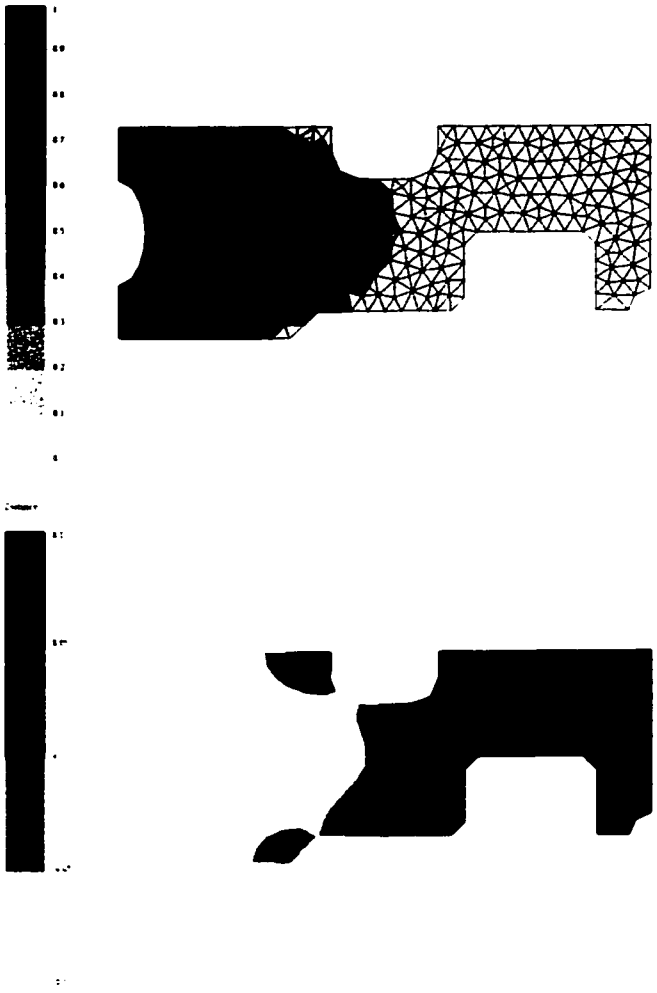


Figure 4.8: Filling factors on isotropic mesh and corresponding distance field

4.3.9 The size map

The size map described here consists of an approximation of the anticipated displacement of the fluid front for the next few time steps. As illustrated in Figure 4.9, a difference exists here between the tangential and normal density factors that govern the element size in the new mesh. This will allow to select separately the element density

factors in the tangential and normal directions (thus the need to create non isotropic elements). In Figure 4.9, a typical density pattern is displayed, where l is simply a parameter of position (taken here along section AA' in Figure 4.5 for example).

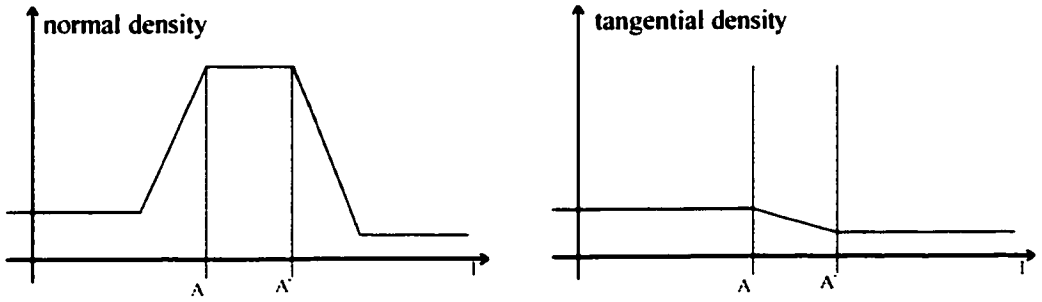


Figure 4.9: Normal and tangential densities along axis AA' (see Figure 4.5).

The tangential size is determined at the beginning and will not change during the whole simulation. The normal size s_n (inverse of the normal density τ_n) depends on the time increment Δt and the number k of finite element layers that will be considered for each calculation step. It is defined relatively to the normal velocity field $\mathbf{V} \cdot \mathbf{n}$ near the moving front. In a local coordinate system defined by the normal to the front and a tangent vector, s_n and s_t can be expressed as follows:

$$s_n = \frac{1}{\tau_n} = \frac{\mathbf{V} \cdot \mathbf{n} \cdot \Delta t}{k} \tag{37}$$

$$s_t = \frac{1}{\tau_t} = cst \tag{38}$$

These equations are valid in the vicinity of the fluid front, i.e., in a layer of thickness e. This thickness e corresponds to the distance (AA') in Figure 4.9 and is calculated as follows:

$$e = (AA') = 3 \cdot \mathbf{V} \cdot \mathbf{n} \cdot \Delta t \tag{39}$$

Anywhere else, except in a transition layer surrounding the refined layer, the mesh will be isotropic with a size factor corresponding to s_t, eventually with some variations from place to place depending on the complexity of the geometry. From these assumptions, the metric **M** used in equations (2) and (3) to define the size map can be computed locally in the global coordinate system (O,x,y) as follows:

$$\mathbf{M} = {}^{T}\mathbf{R} \cdot \begin{bmatrix} \tau_n^{\,2} & 0 \\ 0 & \tau_t^{\,2} \end{bmatrix} \cdot \mathbf{R} \tag{40}$$

Here **R** is a rotation matrix, used to change the expression of the metric field from a local base attached to the front geometry to a global base, and θ is the angle formed by the normal to the front and the direction $\overrightarrow{Ox}$:

$$\mathbf{R} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}, \quad \theta = \left(\overrightarrow{Ox}, \vec{\mathbf{n}}\right) \tag{41}$$

This metric is a quadratic form. With this metric field and the mesh generation algorithm described above, the local mesh refinements of Figure 4.10 have been obtained in a planar mold with obstacles.

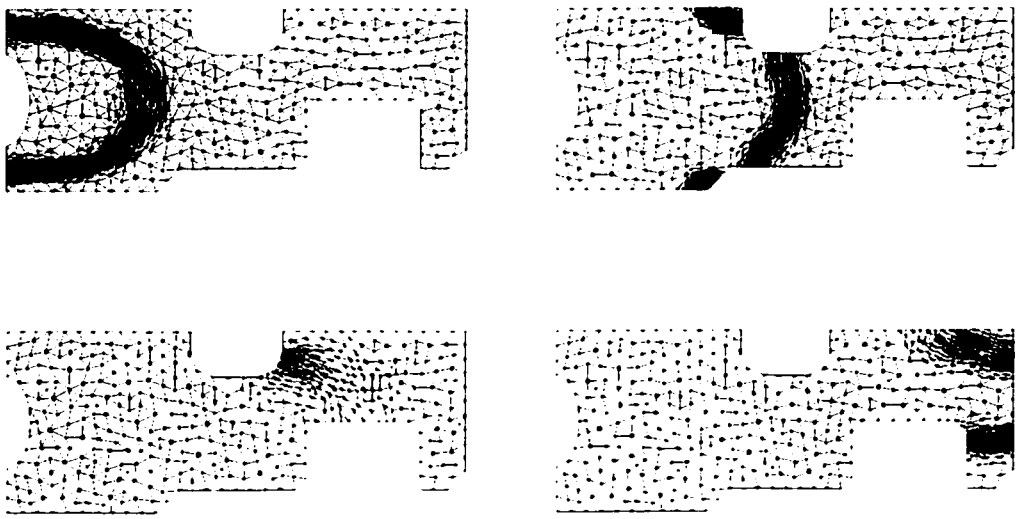


Figure 4.10: Examples of adaptive meshes generated at different time steps in order to follow the moving front.

So an adaptive mesh has been generated, able to follow a moving boundary and respect non isotropic geometrical constraints. Moreover, based on the theory of level sets and on the notion of "distance to the flow front" introduced here, this mesh can be structured to include a given number of finite element layers in the vicinity of the moving boundary. It is now time to include this mesh refinement scheme in the calculation process and implement the algorithm for RTM mould filling simulations.

4.4 Update of the flow front position

The size map introduced in the previous section is used to generate a mesh that will reflect the positions of the moving front. Two meshes have been constructed from the background mesh, one defined at the current time step $t = t_i$ and the other at the next time step $t = t_{i+1}$. The scalar fields of velocity and filling are defined at the current time

step t_i. The location of the flow front at the next time step must be extrapolated in order to update the simulation between two successive time steps. A new notion is introduced for this purpose: the *time field*. It is defined as the time at which a point of the domain is reached by the resin front. Using this information, it is possible to evaluate new filling factors for the next time step and thus, close the simulation loop.

4.4.1 Computation of the time field

The fast marching algorithm mentioned before is used to solve the related Eikonal equation that will lead to the time field. The time is set at 0 on the front. As in equations (5) and (9), $v(\mathbf{X})$ is the normal velocity to the moving boundary at position X (scalar).

$$v(\mathbf{X}) \cdot |\nabla u| = 1 \text{ in } \Omega \text{ (whole domain)} \tag{42}$$

$$u = 0 \text{ on } \Gamma(t_i) \text{ (current front)} \tag{43}$$

The same approach as for the distance field is used to solve the above problem. We compute the arrival time at each node of the domain layer by layer, starting from the front vicinity. One can refer again to figure Figure 4.7 for notations. The time at any node "K" is evaluated by the following recursive formula which replaces equation (10):

$$t_K = (\alpha_K \cdot t_{N_0} + (1 - \alpha_K) \cdot t_{N_1}) + \Delta t_K \tag{44}$$

where

$$\Delta t_K = \frac{|\nabla d|}{\mathbf{V} \cdot \nabla d} \cdot (d_K - (\alpha_K \cdot d_{N0} + (1 - \alpha_K) \cdot d_{N1})) \tag{45}$$

is the time increment in one element. Here d is the distance field, d_k is its value at node "K" and **V** is the velocity vector at this node.

Solving these equations leads to a solution $u = t(\mathbf{X})$, known as the time field, which indicates for any point in the cavity the time of arrival of the resin front. However, one problem remains with this approach. A knowledge of the velocity function **V(X)** is needed in the whole domain Ω. After calculating the current time field, the velocity is known only in the filled region Ω_f. It is therefore necessary to extrapolate the velocity in the vicinity of the flow front from the saturated domain to the empty domain.

4.4.2 Determining the extrapolated velocity field

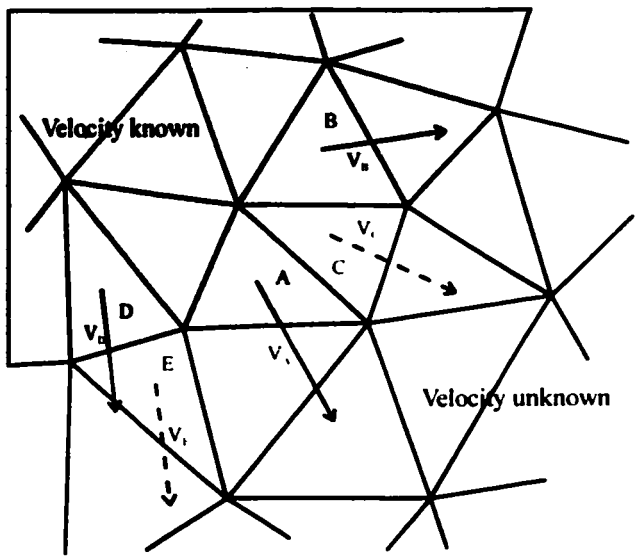


Figure 4.11: Velocity extrapolation scheme

The velocity field is projected layer by layer, starting from elements crossing the current flow front position (0.5 iso-value in the filling field). Jumping from one layer to the other with the help of the distance field (all elements in one layer have approximately

the same distance values), the missing velocity vectors can be generated based on the following heuristics:

(1) When there is only one adjacent triangle, simply translate the velocity vector, for example $\mathbf{V}_E = \mathbf{V}_D$ in Figure 4.11.

(2) When there are two adjacent triangles (triangles "*A*" and "*B*" are adjacent to triangle "*C*" in Figure 4.11), first do :

$$\mathbf{V}_C = \frac{\mathbf{V}_A + \mathbf{V}_B}{2} \tag{46}$$

We need to conserve the norm $|\mathbf{V}_C|$ of the velocity to ensure conservation of the resin mass (for a constant flow front). This can be achieved easily by imposing the following condition:

$$|\mathbf{V}_C| = \frac{|\mathbf{V}_A| + |\mathbf{V}_B|}{2} \tag{47}$$

The direction of vector $\mathbf{V}_C$ remains of course unchanged. It is clear that this approach represents a first approximation and does not ensure a perfect conservation of the resin mass "far" from the flow front. In principle, the resin flow rate should remain constant if integrated along any cross section. This is not the case here, but as the front moves by small increments, the numerical experiments reported in the sequel have shown that fairly accurate results can still be obtained.

4.5 Description of the algorithm

The algorithm of the whole computation writes as follows:

(1) Generate a background mesh "as coarse as possible", which will be used as fixed reference input in the re-meshing algorithm. This mesh must represent accurately the geometry of the part (it can be a 3D surface or curved boundaries can exist), as well as the boundary conditions (Typically, an STL mesh from any CAD software is appropriate here, provided that it can represent boundary conditions that will be set later on)

(2) Generate a finer initial mesh from a size map based on the boundary conditions at the beginning of the injection, then begin the filling calculation for the first time step (can be isotropic and uniform, or the result of an initial adaptation scheme)

(3) LOOP: based on the previous filling factors and injection boundary conditions, calculate with non conforming linear finite elements the new velocity field, as well as the other fields requested (pressure, temperature, etc...).

(4) Save results for this time step.

(5) Calculate the distance and extrapolated velocity fields, then the time field. Based on this, a non isotropic size map is generated, then the new mesh is created from the coarse background mesh by a remeshing technique similar to the one described by the author in [13].

(6) Update the flow front based on a fixed or automatic time step and the time field determined in (5).

(7) Test if the cavity is filled; if not, perform an additional loop by going back to (3).

(8) Display filling results.

4.6 Results

Some comparison has been made, first with standard calculations carried out for a fine mesh (numerical standard), then with experiments in the planar mould with obstacles of Figure 4.13. The cavity is injected by a line injection located on the right side. In order for this two-dimensional test to be as general as possible, the mould includes a zone of convergent flow, i.e. a narrowing channel followed by a zone of divergent flow, and a circular insert. The injection is performed along a line on the right side of the mould in order to obtain a nearly regular flow front at the beginning of the experiment [31].

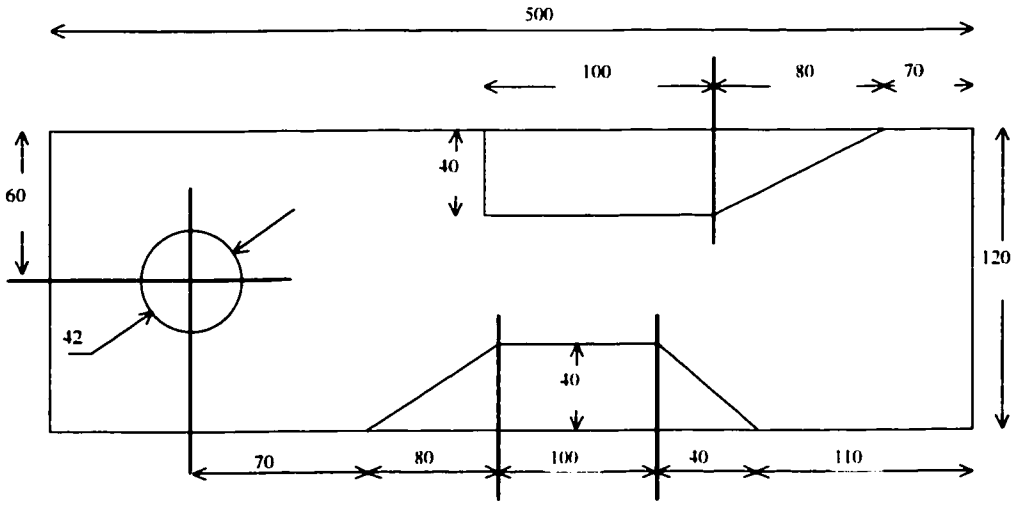


Figure 4.12: Dimensions of the test mould

The experimental results were obtained in the following conditions:

- constant temperature;
- viscosity of the calibrated silicon oil: $\mu = 0.1\ \text{Pa} \cdot \text{s}$;
- permeability of the isotropic fabric: $K = 5.2 \cdot 10^{-10}\ \text{m}^2$ (measured in a separate experiment);

- pressure driven injection: the (non constant in time) injection pressure is measured at the inlet gate and used as boundary condition in the simulation;
- pictures were taken with a digital camera, when the flow front crosses the lines drawn on the glass cover of the mould;
- time was recorded precisely when the fluid front crosses these lines.

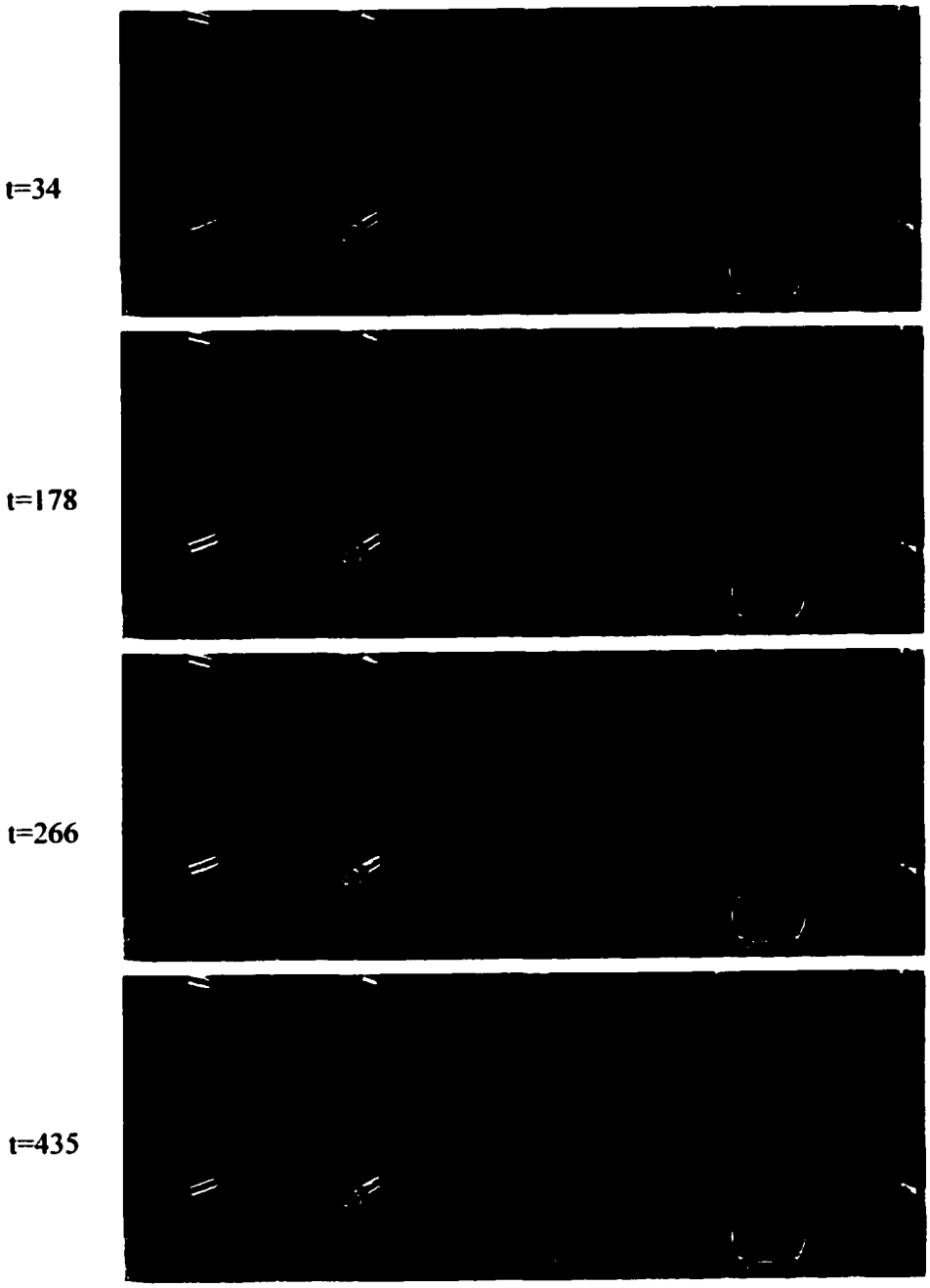


Figure 4.13: Experimental results

Figure 4.14 shows the injection pressure measured during this experience. There are two sudden rises in the curve, as pressure was increased twice to accelerate mould

filling. Note that the constant pressure plateau is not instantaneous, as there is always some delay before a new pressure level is reached. As long as the injection pressure is recorded in time, a simulation based on Darcy's law will reproduce accurately the experimental results of Figure 4.13.

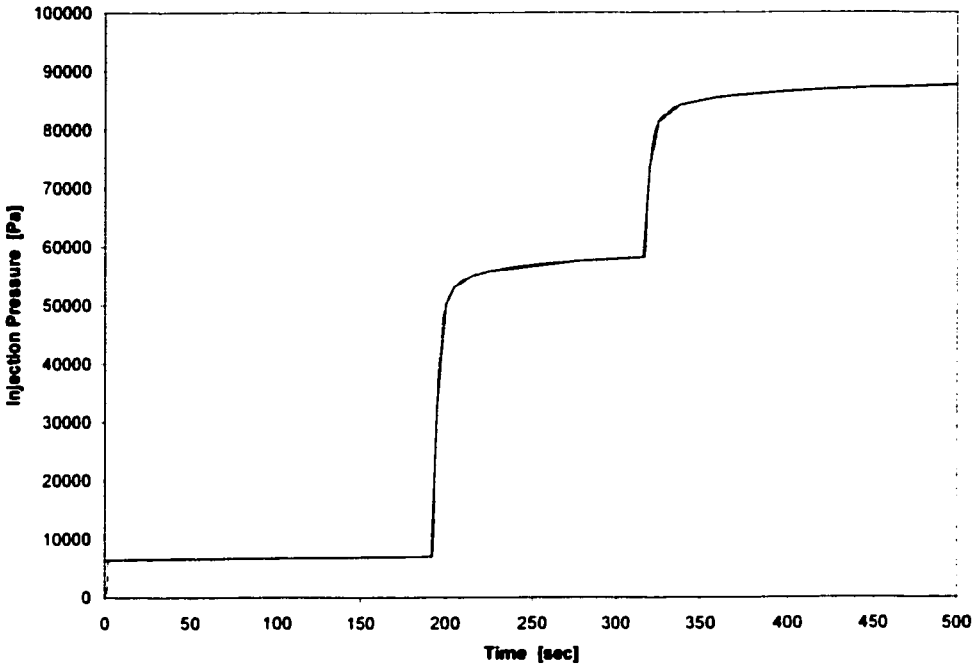


Figure 4.14: Measured pressure at injection gate

Using the same experimental conditions (viscosity, permeability, injection pressure in time), a series of "numerical experiments" was performed by varying the number of calculation steps. Figure 4.15 shows the filling of the mould in time calculated by LCMFlot.

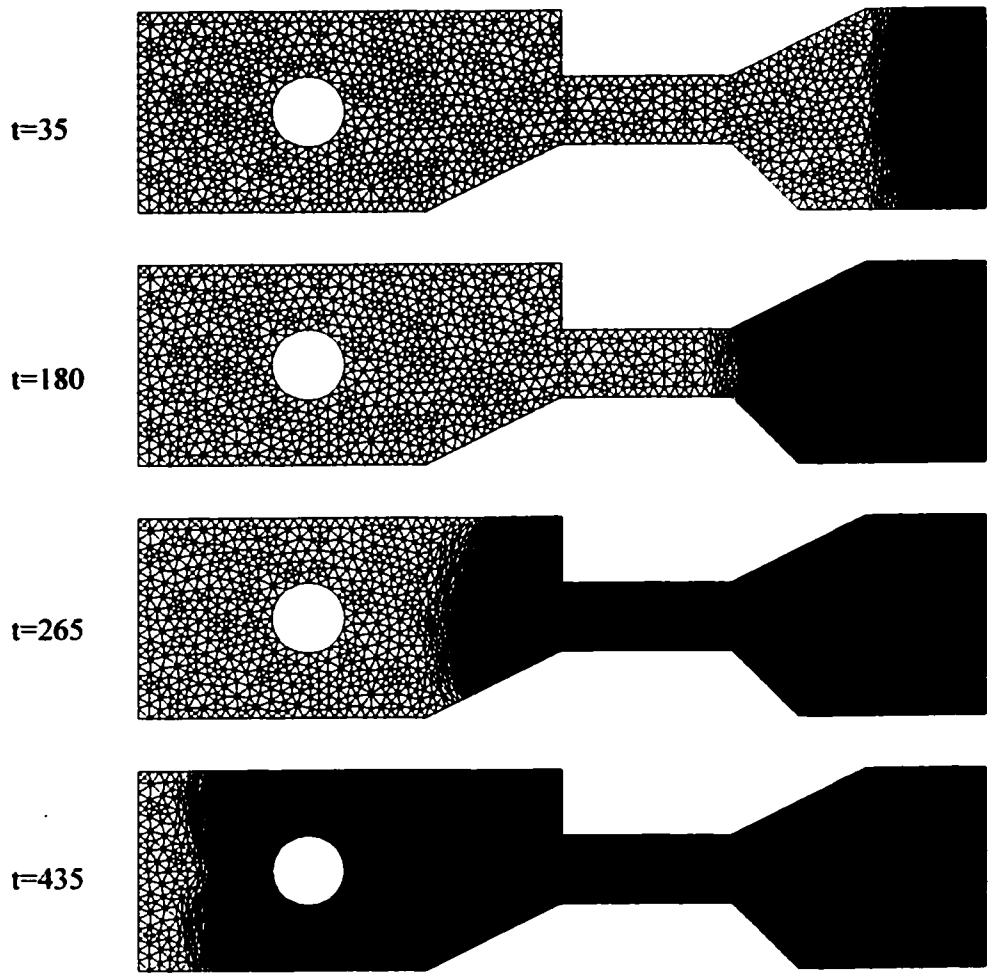


Figure 4.15: Numerical results for a 61 time steps simulation.

Figure 4.16 shows a comparison between the experiment, the numerical simulations with re-meshing and a reference calculation performed by LCMflot with 700 computation steps. This is done on a fixed and highly refined mesh (the approximate element size is half of the size of the elements situated far from the front in Figure 4.15). The flow front position is the position of previously determined points where time is recorded as the front reaches that position. Same points are used in the simulation for that comparison. When compared with the experiment, the reference calculation shows a very accurate validation of the finite element approximation based on Darcy's law to

model mould filling. Re-meshing simulations have also been carried out for an increasing number of time steps from 30 to 300. The results show that the re-meshing algorithm converges rapidly to the reference solution. For 61 re-meshing steps, the approximate positions of the fluid front in time are superposed with the reference solution. Clearly, the re-meshing algorithm allows to reduce the number of calculation steps.

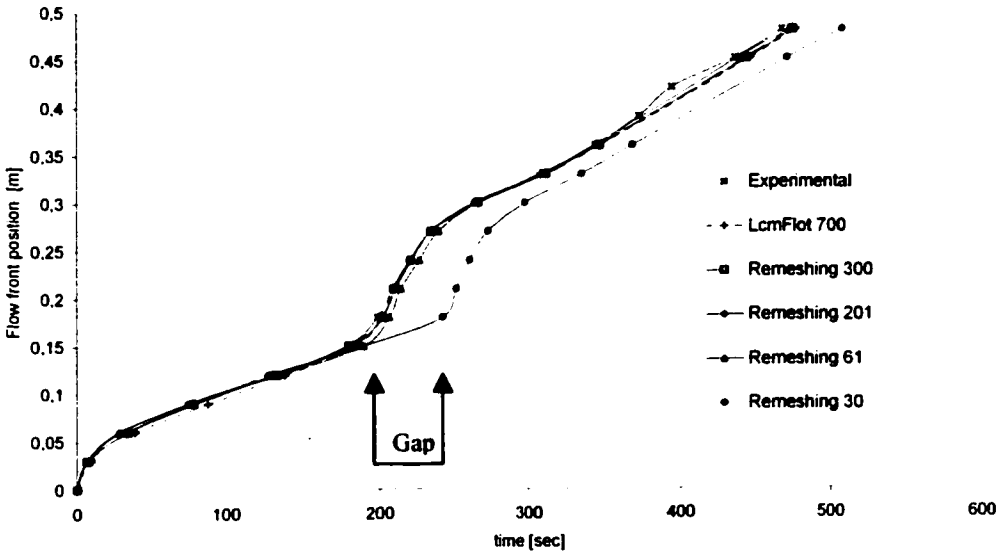


Figure 4.16: Comparison of numerical and experimental results

Note that a gap is observed on the graph between the experiment and the re-meshing calculation for 30 time steps. This can be explained from the shape of the pressure curve. Pressure rises around 200s first, then at 330s. This cannot be accurately taken into account when the number of calculation steps is too low, because it leads to a delay during which the simulation is not yet aware that the injection pressure has risen. In fact, when reducing the number of time steps, one must be aware of the loss of accuracy, especially when there is a sudden change of boundary conditions. This happens in

industrial applications when the pressure or the flow rate is controlled to "optimise" mould filling.

4.7 Conclusion

The numerical results show that a good accuracy can be obtained with the re-meshing algorithm even for relatively few time steps. The shape of the flow front is respected by the re-meshing algorithm, even if the number of degrees of freedom in the finite element formulation remains low. In standard simulations on a fixed mesh, the number of new elements to be filled at each calculation step is closely related to the time increment, because one cannot update the flow front for more then one layer of elements at a time. In fact, in order to ensure a good approximation, in theory only one new filled element should be added to the saturated domain at each calculation step. This kind of constraint means that extensive calculations are required to simulate the progression of the moving front in the mould cavity. Moreover, a very fine mesh is needed to apply precise thermal boundary conditions on the flow front. All these reasons explain why the computational cost of mould filling simulations is usually high.

In conclusion, it is clear that re-meshing allows to reduce significantly the number of calculation steps, each of them including a complete finite element computation of the pressure and velocity fields in the saturated domain. As shown in this paper, this does not hinder the ability to predict accurately the evolution of the flow front in time. Furthermore, although each calculation is performed on a different mesh, the number of degrees of freedom is reduced significantly for each finite element calculation. However, there is a computational cost associated with re-meshing and with the generation of the non isotropic size map. Therefore it is not guaranteed that the computer time will also be reduced in the case of "simple" injection moulding simulations.

4.8 Acknowledgements

We acknowledge the Natural Science and Engineering Research Council of Canada (NSERC), and the Fonds Concerté d'Aide à la Recherche (FCAR) for their financial support.

4.9 References

[12] D. Adalsteinsson, J.A. Sethian, 1999, The Fast Construction of Extension Velocities in Level Set Methods, J. Computational Physics **148**, 2-22.

[13] E. Béchet, J.C. Cuillière, F. Trochu, 2002, Generation of a finite element mesh from stereolithography (STL) files, Computer Aided Design **34**, 1-17.

[14] W. Chang, N. Kikuchi, 1994, An Adaptive Remeshing Method in Simulation of Resin Transfer Molding (RTM) Process, Computer Methods in Applied Mechanics and Engineering, **112**, 41-68.

[15] Y.F. Chen, K.A. Stelson, V.R. Voller, 1997, Prediction of Filling Time and Vent Location for Resin Transfer Molds, J. of Composite Materials, **31** (11), 1141-1161.

[16] C. Cherfils, F. Hermeline, 1990, Diagonal swap procedures and characterizations of 2D-Delaunay triangulations, Math. Mod. And Num. Anal. **24** (5), 613-626.

[17] J.C. Cuillière, 1997, A direct method for the automatic discretization of 3D parametric curves, Comp. Aided Design **29** (9), 639-647.

[18] B. Delaunay, 1934, Sur la sphère vide, Bul. Acad. Sci. URSS, Class. Sci.Nat., 793-800.

[19] G.L. Dirichlet, 1850, Über die Reduktion der positiven quadratischen Formen mit drei unterstimmten ganzen Zahlen, Z. Angew. Math. Mech. **40**(3), 209-227.

[20] P.L. George, H. Borouchaki, 1997. Triangulation de Delaunay et maillages : application aux éléments finis. Paris. Hermès. ISBN 2-86601-625-4.

[21] R. Kimmel, J.A. Sethian, 1998. Fast Marching Methods on Triangulated Domains. Proc. Nat. Acad. Sci. **95**. 8341-8435.

[22] P.P. Pébay. P.J. Frey. 1998. A priori Delaunay-conformity. Proceedings of the 7th International Meshing Roundtable. Dearborn(MI). 321-333.

[23] J. Ruppert. 1994. A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation. Journal of Algorithms **18**. 548-585.

[24] J.A. Sethian. 1996. Fast Marching Level Set Method for Monotonically Advancing Fronts. , Proc. Nat. Acad. Sci.. **93** (4). 1591-1595.

[25] J.A. Sethian. 1999. Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry. Fluid Mechanics. Computer Vision and Materials Science. Cambridge University Press.

[26] F. Trochu. P. Ferland. R. Gauvin. 1997. Functional Requirements of a Simulation Software for Liquid Molding Processes. Science & Engineering of Composite Materials. **6** (4). 209-218.

[27] F. Trochu. R. Gauvin. D.M. Gao. 1993. Numerical Analysis of the Resin Transfer Molding Process by the Finite Element Method. Advances in Polymer Technology. **12** (4). 329-342.

[28] F. Trochu. R. Gauvin. 1992. Limitations of a Boundary-Fitted Finite Difference Method for the Simulation of the Resin Transfer Molding Process. J.of Reinf. Plast. Comp.,**11** (7). 772-786.

[29] G. Voronoï. 1908. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Recherches sur les paralléloèdres primitifs. Journal Reine angew. math. **134**.

[30] D.F. Watson. 1981. Computing the n-dimensional Delaunay Tesselation with application to Voronoï polytopes. Computer Journal **24** (2). 167-172.

[31] W.B. Young, K. Rupel, K. Han, L.J. Lee, M.J. Liou, 1990, Simulation and Experimental Verification of Mold Filling in Resin Transfer Molding and Structural RIM, 45th Annual Conference, Composite Institute, The Society of Plastics Industry.

CHAPITRE 5

ADAPTIVE MESH GENERATION TO SOLVE MOULD FILLING PROBLEMS WITH APPLICATION TO RESIN TRANSFER MOULDING - PART I: TWO-DIMENSIONAL CASE

E. BÉCHET[1,2], E. RUIZ[1], F. TROCHU[1], J.-C. CUILLIERE[2]

(1) Centre de Recherches Appliquées Sur les Polymères (CRASP), Département de Génie Mécanique, École Polytechnique de Montréal, H3C 3A7, Canada
eric.bechet@epost.de; francois.trochu@polymtl.ca

(2) LIRICS, Département de Génie Mécanique, Université du Québec à Trois-Rivières, G9A 5H7, Canada
jean-christophe_cuilliere@uqtr.ca

5.1 Abstract

In injection moulding processes such as Resin Transfer Moulding (RTM) for example, numerical simulations are usually performed on a fixed mesh, on which the numerical algorithm predict the displacement of the flow front. The adaptive algorithm can be used also together with a-priori error estimations to optimise the mesh for the finite element analysis. This optimisation can be implemented during mould filling in order to adapt the mesh to the shape of the moving boundary. However, in order to minimize computer time, it is preferable to optimise the mesh before carrying out the filling calculation. This approach is applied in this paper to resin transfer moulding, a process used to manufacture composites by injection of a polymer resin through fibrous glass or carbon reinforcements: (1) the adaptive algorithm is implemented with error estimations in a mould with obstacles to generate a initial mesh; and (2) non isothermal filling simulations are carried out in a rectangular mould to illustrate the stability conditions that arise from the convective heat transfer problem. An analytical study on the radial injection case is also carried out to illustrate issues related to four types of different mesh

refinement procedures: (1) a mesh allowing for a constant time step, (2) a mesh with constant radial density (thus allowing a constant progression of the flow front at each time step), (3) a mesh providing a constant Courant number, in order to ensure stable thermal simulations; and (4) finally, a mesh for which the interpolation error is constant.

Keywords: Resin transfer moulding, finite elements, thermal analysis, Darcy equation, anisotropic mesh generation, mesh adaptation, error estimator.

5.2 Introduction

Injection moulding simulations are carried out for a variety of industrial processes such as metal casting, thermoplastic injection, liquid composite moulding, etc. This type of analysis requires the combination at each time step of a finite element approximation of the partial differential equation that governs the flow together with a filling algorithm to advance the fluid front in the cavity. This paper is concerned with the application of mesh refinement strategies to the numerical solution of mould filling problems. Although adaptive finite elements have been widely used to solve a large variety of engineering problems, little work has been done on their implementation to moving boundary problems. The numerical approximation of such problems requires generally using a rather fine mesh. If automatic mesh refinement is desired, it should be adapted to the shape of the flow front. A previous article by Bechet et al. [34] has shown how non-isotropic mesh adaptation algorithms can be used efficiently to solve mould-filling problems. This novel approach is discussed here with an application to Resin Transfer Moulding (RTM).

Several computer simulations have been developed in the case of RTM simulations. Bruschke and Advani [36] have combined control volumes with finite elements (FE/CV method) to simulate mould filling in RTM. Young et al. [60] have also developed a

similar numerical model to solve RTM flow problems. Later, Trochu et al. [56] proposed a different approach in which the finite element was also the control volume. In this latter case, using non-conforming finite elements ensures a local conservation of the fluid volume in the cavity. Mould filling calculations are usually costly in terms of computer time, because of the large number of elements required to model the cavity and eventually the mould. This is especially true when non-isothermal analyses need to be carried out. In the work of Lin et al. [49] for example, the heat transfer equation is solved by coupling finite elements in the plane with finite differences through the thickness. Bruschke and Advani [37] have also modelled by finite elements non-isothermal convective mould filling for the RTM process. Trochu et al. [54] has also proposed an algorithm to model non-isothermal problems by Taylor-Galerkin, which was able to reduce, but not eliminate completely the numerical instability that is generally observed in convective heat transfer problems. Non-isothermal RTM flow simulations have been extensively validated by several authors, including Young et al. [59] and Calhoun et al. [39] for mould filling and Guyonvarch et al. [44] for heat transfer. However, a high level of mesh refinement is always necessary to track accurately the motion of the moving boundary in time and a large number of calculation steps is required to model the transient heat transfer as well as the chemical reaction in the case of thermosetting resins.

The numerical schemes used in mould filling simulations are usually based on a time dependent resolution of an unsteady (free surface) boundary value problem. In RTM, Eulerian schemes are generally implemented together with the FE/CV method developed by Maier et al. [46]. The boundary of the filled area in the mould cavity is constantly evolving, and it is difficult to generate an isotropic mesh suitable for all the successive calculation steps of a filling simulation. The fluid front cannot be approximated with a fine precision on an isotropic mesh. Such a mesh would have to be very fine everywhere in the geometrical domain in order to provide an accurate approximation. This leads to relatively time consuming calculations, although a fine

mesh would be required only in the vicinity of the flow front and near the inlet gates. For this reason, several researchers have proposed to construct a new mesh of the fluid saturated domain at each time step (Bechet et al. [34] for Eulerian scheme, Mutin, Coupez et al [50] for Lagrangian schemes). This kind of formulation is long in terms of computer time; it is also rather complex, especially in the case of obstacles, merging flow fronts or three-dimensional problems. A new approach is proposed in this article, based on error estimators calculated during a simulation of the resin flow in the fully saturated cavity. This allows an optimisation of the mesh before launching the filling and thermal calculations and results in improved accuracy and reduced computer time.

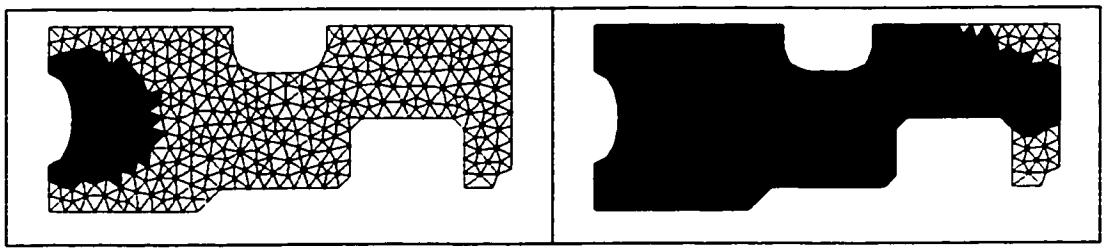


Figure 5.1: Fluid saturated elements (in grey) in a filling simulation on a fixed mesh

Figure 5.1 illustrates a numerical problem connected with mould filling simulations: it is difficult to track a smooth flow front on a fixed mesh. One major difficulty arises from this kind of approximation in non-isothermal resin transfer moulding, when the temperatures of the resin and of the fibre bed are different at the beginning of resin injection. A thermal boundary condition must be specified on the front to solve the heat convection problem during mould filling. Bechet et al. [34] developed an adaptive algorithm capable of generating a new non-isotropic mesh in the vicinity of the flow front. A new filling algorithm was also devised, based on the fast marching approach developed by Sethian [52] and on the theory of level sets developed by Kimmel et al.[45], Adalsteisson et al. [32], and Sethian [53]. By implementing this approach, the number of calculation steps could be reduced considerably, while preserving the same overall accuracy of the numerical calculations. This works well for quasi-static

problems, when there is no functional dependence between the time steps. However, as discussed in this article, it becomes less easy to implement when numerical instability problems arise in convective thermal simulations. The time step must be reduced significantly (at least in the thermal analysis). This decreases considerably the gain that can be expected from adaptive meshing algorithms.

These problems can be avoided when an a-priori adapted fixed mesh is considered. Most existing software packages rely on a fixed mesh strategy to simulate mould filling. Some are based on the control volume approach. For example, Bruschke and Advani [36] have adopted a node-based representation of filling factors. Trochu et al. [56] followed a different approach and implemented an element-based representation of filling factors. When adaptive procedures are used, remeshing is usually performed for each new position of the flow front, which makes the simulation quite expensive in terms of computer time. For example, Chang and Kikuchi [40] have used a remeshing procedure to simulate the RTM process. This work was focused on the numerical solution of Darcy's equation at each time step in the saturated domain, but no assumption was made on the accuracy of the flow front. This paper proposes a new way to improve at the same time the accuracy and efficiency of mould filling simulations. The adaptive algorithm is used here to generate a pre-optimised mesh that will take into account the geometry of the part and the material properties of the reinforcement in order to decrease numerical errors. The filling simulation is carried out on a fixed, but adapted mesh, with all the computational advantages derived from this approach. The shape of the flow front is rendered with much more accuracy with an adapted mesh. The resulting gains include an improved stability of thermal simulations and a reduced computer time required to carry out the filling calculations. For uniform filling of cavities with no time dependence of the injection conditions, Chen et al. [41] proposed a one-shot injection prediction. This represents a good alternative to time-explicit simulations. However, when non-linear phenomena are taken in account (such as in

curing analyses for example), the need for several convergence loops does not make this approach more computationally efficient than usual time-explicit simulations.

In Part I of this contribution, the adaptive algorithm is presented in the two-dimensional case. The main novel features of this investigation are the following: (1) a-priori error estimations are implemented in a mesh generation algorithm to optimise isothermal RTM mould filling calculations (in particular, non-isotropic material properties of the fibrous reinforcement are taken into account); (2) adaptive meshing is also applied to non-isothermal RTM flow analysis; (3) finally, different ways of creating optimal meshes are illustrated and discussed for the RTM process in the case of a radial injection. Part II expands these concepts to three-dimensional shell geometries, which represent the most common examples of composite parts currently made by resin injection through fibrous reinforcements (RTM process).

5.3 Surface mesh generation

An initial mesh must be provided, that must be fine enough to represent the geometry of the part, typically a STL file most of the time. A standard bisection algorithm is used to refine the original mesh. This methodology was described in Béchet et al. [33] for the isotropic scheme. It is briefly summarized in the next subsection in the isotropic and non isotropic cases.

5.3.1 Isotropic scheme

New vertices are generated by bisection of a line segment already existing in the mesh. A new vertex is created near the middle of the longest line segment (hence increasing the density of the mesh). This vertex must be inserted topologically and projected on the initial mesh. After each bisection, the mesh must be modified locally in order to achieve

a better regularity in the size of the elements. This is done by respecting the well-known Delaunay criterion for 2D meshes [42] [51].

5.3.2 Non isotropic extension

As we need here to deal with anisotropy, the refinement algorithm must be adapted to generate a non isotropic mesh. This can be achieved by constructing a non isotropic size map that will indicate how stretched triangles should be everywhere on the surface. A new notion of distance will be associated to this non isotropic size map. The refinement algorithm described above in the isotropic case will produce a non isotropic mesh when implemented with this new distance. This information on the desired size of the elements needs to be provided in two orthogonal directions. The orientation of these directions must also be specified. In two-dimensions, three independent parameters are required to define a non isotropic size map. This can be summarized with a 2x2 symmetric positive definite matrix, called the *metric*, or a similar metric field function of position if the element size is not constant over the computational domain:

$$\mathbf{M}(\mathbf{X}) = \begin{bmatrix} a(\mathbf{X}) & c(\mathbf{X}) \\ c(\mathbf{X}) & b(\mathbf{X}) \end{bmatrix} \tag{48}$$

The notion of distance between two points is then redefined as follows [43]:

$$dist(AB) = l(\Gamma) = \int_0^1 \sqrt{{}^t\left(\frac{\partial \mathbf{s}(t)}{\partial t}\right) \cdot \mathbf{M}(\mathbf{s}(t)) \cdot \left(\frac{\partial \mathbf{s}(t)}{\partial t}\right)}\, dt \tag{49}$$

where A and B are the points considered, $\mathbf{s}(t)$ is a parametric representation of the path followed on the surface to go from A to B (usually a straight line) and $\mathbf{M}$ is the metric field. The desired size at every position depends on the metric field. In fact, when

evaluated with the proposed metric, the length of any line segment of the resulting mesh should be close to one. The longest segments (with respect to the size map in the metric field) are to be refined first.

In our work, this distance is approximated by Simpson's integration scheme. However, if the metric is locally constant along the line segment AB, the following simplified quadrature formula can be used:

$$dist(AB) = \frac{\sqrt{{}^t\overrightarrow{AB}\cdot\mathbf{M}(A)\cdot\overrightarrow{AB}} + \sqrt{{}^t\overrightarrow{AB}\cdot\mathbf{M}(B)\cdot\overrightarrow{AB}}}{2} \tag{50}$$

Of course, the above expression is faster to compute, but less accurate if there are great variations in the metric. This formula cannot be used at the beginning of the algorithm, because line segments of the background mesh cross all the domain (and the moving boundary). Note that the numerical scheme may degenerate if these distances are not calculated with enough accuracy. To overcome this, an automatic adaptive integration scheme is used. It adapts the discretization step along the path to the variations of the metric and avoids the degeneracy of the mesh generation algorithm.

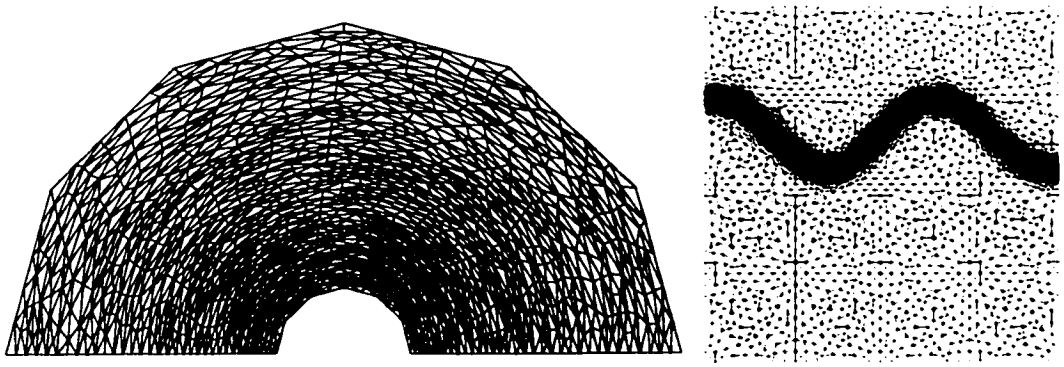


Figure 5.2: Two anisotropic meshes generated with analytical metrics

5.4 Error estimation

5.4.1 Anisotropic error estimator and metric

An error estimator will be described here for discrete three dimensional continuous surfaces, composed of triangular patches. Recently, mesh adaptation became popular with the ability to use *a posteriori* error estimations based on the energy related to each element, and/or gradients of higher order of the solution field. This allowed to solve complex problems (in terms of numerical complexity) with a good quality versus computer time ratio. This approach reduces also to a great extent the number of degrees of freedom needed for each time step in non stationary problems. In that specific case, the adaptation can be made from one calculation step to the next without requiring a full convergence of the adaptive procedure for each iteration. If the time step is short enough, it is not a bad approximation to take the error computed previously to determine the new mesh for the current time. This leads to an economy of additional convergence loops, and hence of computer time. This is valid only if the problem does not show discontinuities in time, which is usually the case in resin transfer moulding. However, if this happens, the strategy is still valid if one uses a time stepping scheme able to adapt to steep variations of the conditions of the simulations (boundary conditions among others).

The error estimator considered here is based on the second derivatives of the primal variable of the finite element formulation (pressure p for a Darcy flow, temperature T for a heat transfer problem, etc..). The Hessian matrix that will act as error estimator is defined as follows in a two-dimensional space:

$$\mathbf{H}(p)=\begin{bmatrix} \frac{\partial^2 p}{\partial x^2} & \frac{\partial^2 p}{\partial y \partial x} \\ \frac{\partial^2 p}{\partial x \partial y} & \frac{\partial^2 p}{\partial y^2} \end{bmatrix} \tag{51}$$

Except at material boundaries (where there is a sudden change of physical properties), the field p is continuous and derivable in the physical space, making this matrix symmetrical. From the Hessian matrix, it is easy to define a related metric. However, its eigenvalues must be all strictly positive in order to derive a distance from it. The Hessian matrix is dependent on the sign of the second derivatives, whereas a metric just needs to know the "size" of elements (obviously positive numbers) in two different directions. The transformation here consists of finding eigenvalues and eigenvectors $\{\mathbf{v}_i\}$ of the Hessian matrix, which becomes then diagonal :

$$\mathbf{\Lambda}=\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \tag{52}$$

We denote $\mathbf{E} = \{\mathbf{v}_i\}$ the eigenvectors such that :

$$\mathbf{H}={}^{\mathrm{T}}\mathbf{E}\cdot\mathbf{\Lambda}\cdot\mathbf{E} \tag{53}$$

By replacing the eigenvalues by their absolute values, we have :

$$|\mathbf{\Lambda}|=\begin{bmatrix} |\lambda_1| & 0 \\ 0 & |\lambda_2| \end{bmatrix} \tag{54}$$

The following metric can be defined :

$$\mathbf{M}(p)=^{\mathrm{T}}\mathbf{E}\cdot|\Lambda|\cdot\mathbf{E}\cdot\frac{1}{\varepsilon} \tag{55}$$

This metric will drive the mesh generation algorithm to produce a mesh that will lower the absolute interpolation error at a constant value ε over the whole computational domain. Of course, the factor ε can be tuned also to match a constant relative error (relative to the value itself of the unknown variable, p , or of its first derivative $\|\nabla p\|$, depending on the application).

5.4.2 Evaluation of the Hessian matrix for linear finite elements

The evaluation of $\frac{\partial^2 p}{\partial x_i \partial x_j}$ cannot be made separately over each element. In our case, the finite element interpolant is linear, so the second derivatives vanish inside an element. Instead, information from neighbouring elements is used to define a local approximation used to recover the second derivatives. In two references [38][47] this approximation is made with the help of Galerkin method to evaluate the first derivatives at each node k of the triangulation (E_j is the j^{th} element around node k, and V_k is the volume of that element):

$$\left.\frac{\partial p}{\partial x_i}\right|_k \approx q_{x,k} = \frac{\sum_j V_j \left.\frac{\partial p}{\partial x_i}\right|_{\mathrm{E}_j}}{\sum V_j} \tag{56}$$

It is then easy to calculate a piecewise constant approximation for each element E_o of the mesh with the help of the inverse $\mathbf{J}^{-1}$ of the Jacobian matrix used to change coordinates in the element considered:

$$\left.\frac{\partial^2 p}{\partial x_i \partial x_j}\right|_{E_o} = \mathbf{J}_{E_o}^{-1} \cdot \left.\frac{\partial\left(\frac{\partial p}{\partial x_i}\right)}{\partial u_k}\right|_{E_o} \tag{57}$$

$$\left.\frac{\partial\left(\frac{\partial p}{\partial x_i}\right)}{\partial u_k}\right|_{E_o} = \frac{\left.\frac{\partial p}{\partial x_i}\right|_k - \left.\frac{\partial p}{\partial x_i}\right|_l}{u_k|_k - u_k|_l} = \left.\frac{\partial p}{\partial x_i}\right|_k - \left.\frac{\partial p}{\partial x_i}\right|_l \tag{58}$$

for a linear interpolation over the element considered. Subscript "*l*" stands for the node defining the origin of the local coordinates in the element, and "*k*" denotes the other local nodes of the element.

5.4.3 Application to mould filling simulations

After having defined the error estimator, an adaptive finite element procedure can be constructed to solve mould filling problems as follows :

i) A fixed mesh is used for the whole simulation.

ii) At each time step, a solution of the flow problem is found.

iii) This solution is used to update the filled region and advance the flow front for the next iteration.

The accuracy of the simulation depends on the initial mesh used to calculate the solution of each time step. The filling algorithm considered here [55] depends also on the mesh. The size of the elements determines in fact the duration of the time step, because at least one new element must be filled up in order to move the flow front for the next iteration. Therefore, the new mesh generated by the adaptive algorithm must be suitable not only

to solve the flow problem, but also to approximate the flow front. In addition, as shown later, the mesh should respect specific conditions on the time step or the spatial discretization if it will be used to perform a thermal analysis with transport terms as well. In the latter case, there are restrictions due to the CFL condition.

5.4.4 Application to Darcy's law

Darcy's law govern flows in porous media. It is an elliptic PDE which is expressed from the conservation of mass:

$$\mathbf{v}_d = -\frac{\mathbf{K}}{\mu} \cdot \nabla p \tag{59}$$

where :

$\mathbf{K}$	is the permeability tensor of the porous medium (m^2)
μ	is the cinematic viscosity of the fluid ($pa \cdot s$)
p	is the total pressure (pa)
$\mathbf{v}_d$	is Darcy's velocity ($m \cdot s^{-1}$)

For an incompressible fluid of constant density and a steady flow without dynamic effects as in Darcy's case, the conservation of the fluid mass implies the divergence identity:

$$\nabla \cdot \mathbf{v}_d = 0 \tag{60}$$

which yields when combined to Darcy's law (59) an elliptic partial differential equation:

$$\nabla \cdot \left(\frac{\mathbf{K}}{\mu} \cdot \nabla p \right) = 0 \tag{61}$$

for an uncompressible fluid and a steady flow without dynamic effects. The effective velocity $\mathbf{v}_e$ $(m \cdot s^{-1})$ of the fluid or average particle velocity is connected with Darcy's velocity though the local porosity of the reinforcement, ϕ (%) by the relation :

$$\mathbf{v}_e = \frac{\mathbf{v}_d}{\phi} \tag{62}$$

In fact, $\mathbf{v}_e$ represents also the velocity of the fluid front for particles near the front. It is larger than Darcy's velocity because the porosity ϕ is always smaller than one.

The equation is solved at each time step using non conforming linear finite elements on triangles or tetrahedron, which are chosen for their ability to conserve of the resin flow rate along inter-element boundaries (for a discussion about this choice, see [56]). Equations (59) and (61) yield the velocity and pressure in the saturated domain at each time step. The effective velocity is used to update the front position. The filling algorithm determines the time increment needed to fill up completely at least one new element, then the boundary condition is updated and the flow front is advanced for the next iteration.

5.4.5 Adaptive algorithm

Adaptation of the mesh is made using a convergent adaptation loop. This is done by calculating the solution of the flow in a completely filled cavity as many times as necessary, until convergence of the numerical scheme and low interpolation error. In the

loop, once the solution of the flow is known, the mesh for the next iteration is generated. The last adapted mesh obtained will be used during the simulation of the "true" injection process. The aim is to improve "*a priori*" a fixed mesh by performing a small series of simulations on a completely filled cavity. Thus, we must define a numerical error that will drive the mesh refinement algorithm.

The error considered here is based on the norm of the Hessian matrix of the pressure field. This norm defines an error estimator suitable for linear finite elements. A metric is then defined as follows:

$$\mathbf{M}(p) = \frac{1}{\varepsilon} \cdot |\mathbf{H}(p)| \tag{63}$$

Recall here that the notation $|\mathbf{H}(p)|$ refers to the same matrix, but with the absolute value of its eigenvalues as defined in equation (55). This metric is generally anisotropic. If the reinforcement is isotropic, there is no need for anisotropic mesh generation as the diffusion of the fluid in the cavity is generally isotropic. For that purpose, the following metric will be used:

$$\mathbf{M}_\varepsilon(p) = \frac{1}{\varepsilon} \cdot \mathbf{I} \cdot \max_i(|\lambda_i|) \tag{64}$$

In the above equation, λ_i are the eigenvalues of $\mathbf{H}(p)$, and $\mathbf{I}$ is the identity matrix.

The user provides an upper bound ε of the error, which controls the adaptation algorithm in order to generate a mesh that will guarantee a given accuracy in the filling simulation. It is assumed that the upstream flow does not depend much on the position of the front. For example, this means that we admit that the flow around a sharp edge

will exhibit the same behaviour during the complete filling of the mould. If this hypothesis is acceptable, a fixed adapted mesh can be used advantageously to lower the interpolation error.

However, the problem involves also the convection of chemical species and heat when non isothermal injection conditions are set. Zaki [61] found that the stretching of elements along the flow directions may be useful to lower the numerical diffusion brought by various numerical schemes (Lesaint-Raviart in particular). Therefore, the metric defined above for a strictly diffusive problem must include some information regarding the hyperbolic aspect of the true physical problem. Besides the "elliptic" metric of equation (63), a new "hyperbolic" metric, based on the local velocity v of the flow, is introduced below :

$$\mathbf{M}_h(\mathbf{v}) = {}^{\mathrm{T}}\mathbf{R} \cdot \begin{bmatrix} \dfrac{1}{\|\mathbf{v}\|^2} & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \dfrac{1}{\alpha^2} & 0 \\ 0 & \dfrac{1}{\beta^2} \end{bmatrix} \cdot \mathbf{R} \tag{65}$$

where **R** is a rotation matrix:

$$\mathbf{R} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{66}$$

In this matrix, θ is the oriented angle between the velocity and the Ox axis. It is determined by its sine and cosine:

$$\cos\theta = \frac{\overrightarrow{Ox} \cdot \mathbf{v}}{\|\mathbf{v}\|} \tag{67}$$

$$\sin\theta = \frac{\left\| \overrightarrow{Ox} \wedge \mathbf{v} \right\|}{\|\mathbf{v}\|} \tag{68}$$

The constants α and β in equation (65) are respectively a scale factor related to the norm of the velocity (it represents a time step between two layer of elements) and the density of elements in other directions than the velocity vector. They can be manually set or determined by an adequate global error estimator. In particular, α can be set easily as it has a very explicit meaning. The factor β can be set to fit many purposes. One can set it to get an isotropic mesh, or to meet specified anisotropy factor. Also, it can be set directly by an error estimator.

A combined metric $\mathbf{M}_c(p, \mathbf{v})$ suited for the whole problem (diffusive for the Darcy problem, and convective for the evolution of the front) is constructed to take the highest mesh density in each direction based on the metrics $\mathbf{M}_h(\mathbf{v})$ and $\mathbf{M}_c(p)$. Using the eigenvalues of $\mathbf{M}_h(\mathbf{v})$, it is easy to reconstruct a new metric with these requirements. Obviously, the matrix composed of the eigenvalues of $\mathbf{M}_h(\mathbf{v})$ is taken from its expression in equation (65) :

$$\Lambda = \begin{bmatrix} \dfrac{1}{\|\mathbf{v}\|^2} & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \dfrac{1}{\alpha^2} & 0 \\ 0 & \dfrac{1}{\beta^2} \end{bmatrix} = \begin{bmatrix} \dfrac{1}{\|\mathbf{v}\|^2 \cdot \alpha^2} & 0 \\ 0 & \dfrac{1}{\beta^2} \end{bmatrix} \tag{69}$$

Thus, the expression of the compound metric becomes :

$$\mathbf{M}_c(p,\mathbf{v}) = {}^{\mathrm{T}}\mathbf{R}\cdot\begin{bmatrix} \max\left(\dfrac{1}{\|\mathbf{v}\|^2\cdot\alpha^2}, \dfrac{1}{\varepsilon}\max_i(|\lambda_i|)\right) & 0 \\ 0 & \max\left(\dfrac{1}{\beta^2}, \dfrac{1}{\varepsilon}\max_i(|\lambda_i|)\right) \end{bmatrix}\cdot\mathbf{R} \qquad (70)$$

In addition, a global density factor η_i is defined for each loop of the algorithm, which will be used to control the adaptive algorithm. This factor prevents the generation of meshes that are over-refined at the early stages of the remeshing loop. It prevents also the oscillations of the solution while converging to the adapted mesh. The metric is modified as follows :

$$\mathbf{M}_c^{\eta_i}(p,\mathbf{v}) = \frac{\mathbf{M}_c(p,\mathbf{v})}{\eta_i} \qquad (71)$$

At each step, η_i is modified to fit with the increase of accuracy in the solution (and increase in the accuracy of the error estimator with regard to the exact solution). The initial value η_0 is given (we have chosen 5 in the latter example), but this parameter may be defined also by users. A good value is in the range between 5 and 15. Then, the following updating scheme is implemented:

$$\begin{cases} \eta_{i+1} = \dfrac{\eta_i}{2} & \text{if } \eta_i > 2 \\ \eta_{i+1} = 1 & \text{if } \eta_i \le 2 \end{cases} \qquad (72)$$

5.4.6 Results of the mesh adaptation

The precedent formulas were implemented to drive a mesh adaptation algorithm on a planar geometry with some features requiring locally a more refined mesh : sharp angles, narrowing channel, obstacle etc.. The adaptation algorithm used here gradually increases the overall mesh density factor until the desired accuracy is reached. This prevents to over-refine the mesh in the first iterations when the solution is not yet accurate enough. Figure 5.3 shows the results of a calculation carried out for a desired maximum relative error of 0.35 .

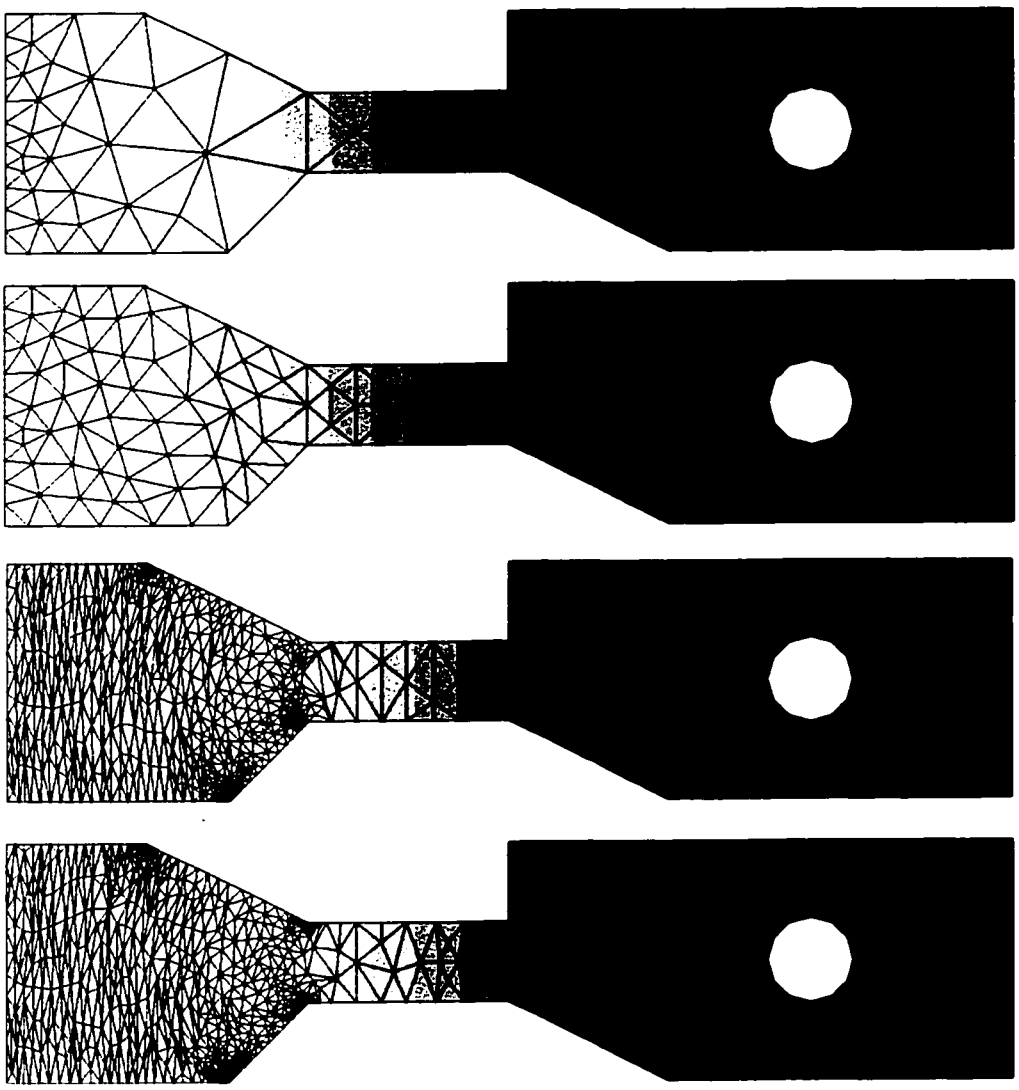


Figure 5.3: Several stages of mesh adaptation are carried out for a steady Darcy flow simulation. Shaded bands show the variations of the pressure field. The inlet is on the left side, outlet on the right

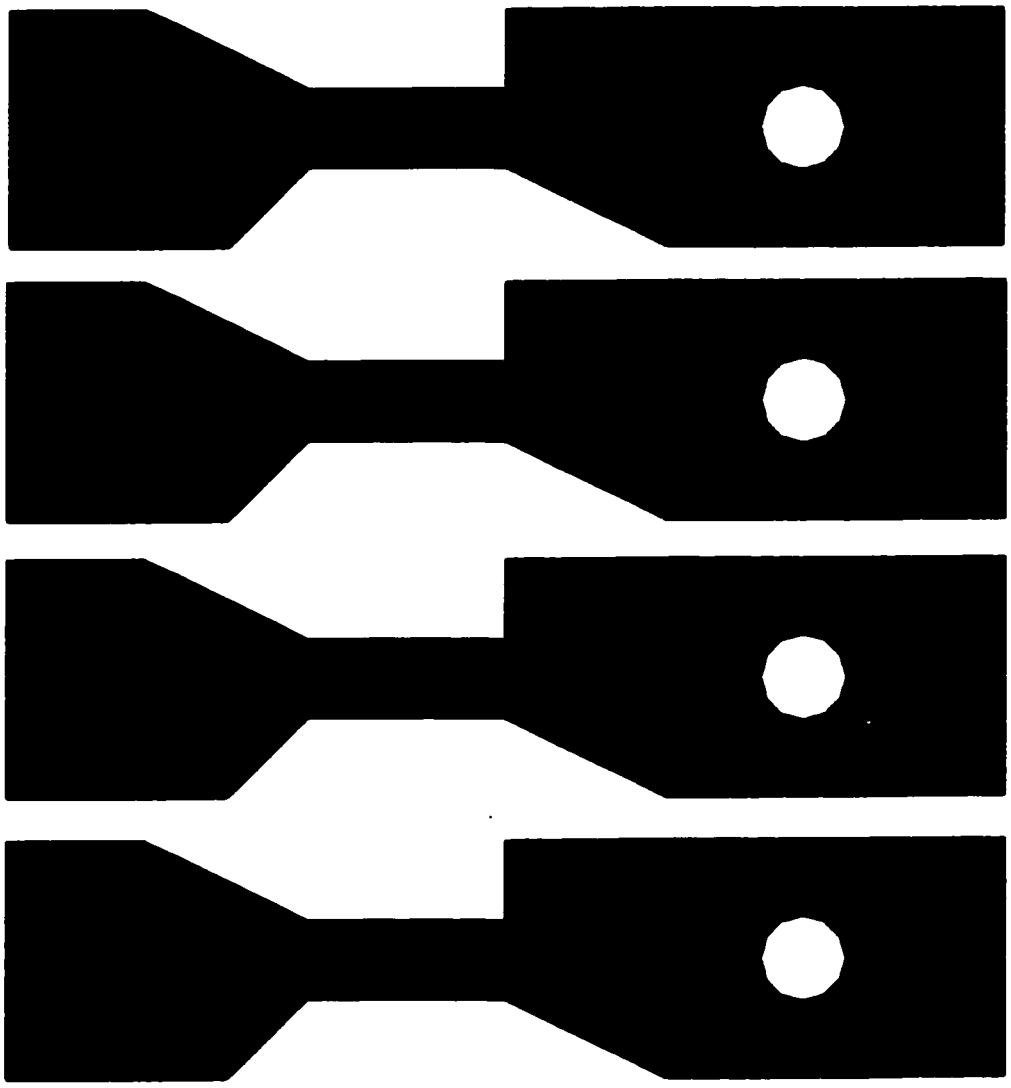


Figure 5.4: Based on the adapted mesh obtained in the last picture of Figure 5.3, a filling simulation was carried out. The results are shown at time 121, 242, 363 and 426 seconds, respectively. The inlet is on the left side, outlet on the right.

5.5 Remeshing for thermal analysis

The remeshing procedure of Bechet et al. [34] was developed originally with the goal of improving the thermal analysis for RTM simulations. The main idea was to control the

numerical diffusion near the flow front. In fact, numerical methods tend to add in heat convection problems a fictive diffusion to the physical problem. Tucker [58] has expressed this diffusion by considering the following parameter:

$$\alpha_{grid} = \frac{\|\mathbf{v}\| \cdot \Delta x}{2} \tag{73}$$

which is added up to the governing equations in the isotropic case as a term like :

$$D_{num} = \alpha_{grid} \cdot \nabla^2 T \tag{74}$$

One way to decrease this diffusion is to decrease Δx, the size of the elements. Ideally, one can decrease it only in the direction of the flow, thus generating anisotropic finite elements. The advantages of adaptive finite elements remain the same : reduction of computer time, and increase or at least control of the accuracy of the numerical simulation. However, the implementation of adaptive remeshing strategies is not straightforward. The successive time steps of a thermal simulation are inter-dependent. This implies a re-interpolation of the solution fields (temperature, degree of cure) in the same iteration loop, while a simple Darcy problem does not have this requirement. As one can expect, this may add again numerical diffusion to the original problem. This is addressed in out implementation by avoiding to change the whole mesh from one time step to another. Thus, it keeps the diffusion as low as possible in most of the computational domain. Another problem arises from the use of remeshing. Much longer time steps are allowed which increase the Courant and Fourier numbers. This leads eventually to numerical instabilities in the algorithms used to solve the convection-diffusion problem. These concerns are addressed in the sequel using a different time step for thermal simulations and Darcy's flow simulations.

After recalling the heat equation for a porous flow, the remeshing algorithms will be implemented for a thermal analysis carried out for two model problems : (1) a line injection in a rectangular mould; (2) a radial injection in a circular mould. Since many RTM injections are carried out from a circular inlet port, it is appropriate also to analyse the behaviour of the remeshing algorithm for a radial injection. These two configuration of injection ports are, with the peripheral injection, the major ways of injecting a RTM part (an example of peripheral injection is presented in part II).

5.5.1 Thermal problem

The original thermal formulation considered here is the following [35][57] :

$$\left(\langle \rho c_p \rangle\right)\frac{\partial \langle T \rangle}{\partial t} + \rho_f c_{p_f} \langle \mathbf{v} \rangle \cdot \nabla \langle T \rangle = \nabla\left(\mathbf{k} \nabla \langle T \rangle\right) - \rho_f \Delta H \frac{d\langle \chi \rangle}{dt} - \langle \mathbf{v} \rangle \nabla \langle p \rangle \tag{75}$$

where

subscript "f" denotes "fluid".

$\langle \ldots \rangle$ denotes a spatial averaging in the control volume.

T is the temperature(K),

P is the pressure (Pa).

$\mathbf{v}$ is the velocity vector of the fluid ($m \cdot s^{-1}$).

χ is the reaction factor of the resin (%).

$\mathbf{k}$ is the effective thermal conductivity tensor ($J \cdot K^{-1} \cdot m^{-1}$).

ρ is the density ($kg \cdot m^{-3}$).

c_p is the calorific capacity($J \cdot K^{-1} \cdot kg^{-1}$).

ΔH is the enthalpy of reaction of the resin $(J \cdot kg^{-1})$

The numerical scheme used to solve the thermal formulation is a standard Galerkin method [35] for diffusion and Lesaint-Raviart [48][61] for convection. The time discretisation used here is Euler's scheme for diffusion (Galerkin), and Gear's implicit scheme for convection (Lesaint-Raviart). The choice of those scheme is discussed in [35]. For the sake of simplicity, the source term arising from the chemical reaction $\rho_f \Delta H \frac{d\langle \chi \rangle}{dt}$ is not considered (i.e., there is no chemical reaction), and the viscous dissipation term $\langle \mathbf{v} \rangle \nabla \langle p \rangle$ is negligible when compared to diffusion and transport. This leaves us with a simplified formulation of the heat equation, the parameters a, b and $\mathbf{k}$ being obviously related to the physical constants of equation (75) :

$$a\frac{\partial T}{\partial t} + b \cdot \mathbf{v} \cdot \nabla T = \nabla(\mathbf{k} \cdot \nabla T) \tag{76}$$

5.5.2 Line injection in a rectangular mould

Thermal simulations are carried out of a line injection in a rectangular mould, and are compared in the following cases: (1) fixed mesh; (2) adaptive remeshing: (3) fixed mesh with larger time steps; (4) modified remeshing algorithm. The adimensional Courant number governs the stability of the numerical algorithm. By respecting some conditions on the time step used in the iterative calculations, it is possible to ensure stability.

5.5.2.1 Fixed mesh simulation

A standard simulation in a simple geometry (L=1m, l=0.1m) was performed with Lcmflot, for the parameters a=b=2,01.$10^6 J \cdot K^{-1} \cdot m^{-3}$ and $\mathbf{k}$=0.25 $J \cdot K^{-1} \cdot m^{-1}$ in

equation (76). At each time step, the velocity v is computed before the thermal problem using a Darcy formulation with non conforming linear finite elements, see equation (61). Resin is injected from the left side; a vent is located at the right side. The other conditions of the simulation stand as follows :

p_{inj}	(injection pressure) :	$2{,}0.10^5\ Pa$.
T_{inj}	(temperature of the injected resin) :	340 K, except at the beginning
μ	(resin viscosity) :	$0{,}1\ Pa \cdot s$.
K	(permeability of the fabric) :	$10^{-9}\ m^2$.
T_w	(temperature of the mould walls) :	400 K.
T^f_{init}	(initial temperature of the fibres) :	340 K.

It must be mentioned here that the initial temperature of the fibres has little effect on the simulation, because we deliberately choose a very low thermal capacity factor for the fibres. Along with that, the fibre content is less than 1%. These assumptions have been chosen to help clarify the effect of transport on the simulation (i.e., there is not much influence on the fibre bed on the resin evolution). The injected resin shows a sudden drop in temperature just at the beginning of the injection. It helps to "destabilize" the numerical scheme, but it has little effect on the final injection results. The results are shown in Figure 5.5 for a few time steps. There are 254 time steps, so the average time step is around 1 s.

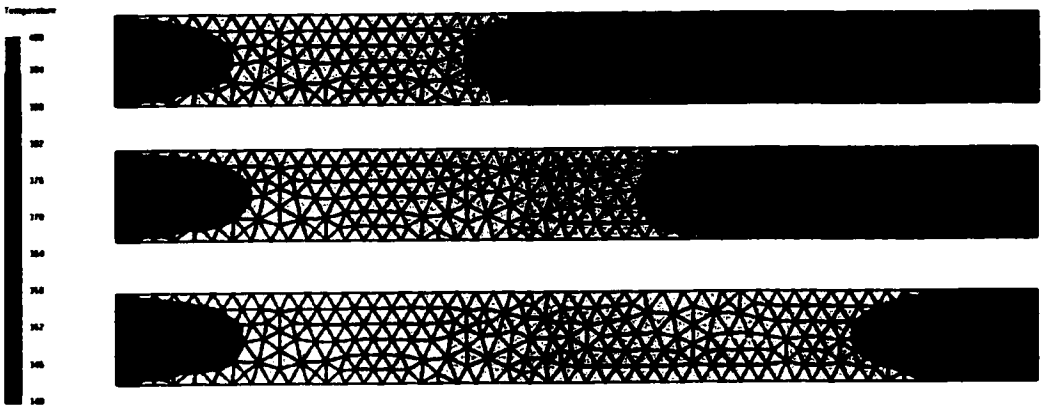


Figure 5.5: Temperature calculated with a fixed mesh, taken at 50*s*, 100*s* and 200*s* after the beginning of injection

As one can expect with such condition, the flow is fully developed and a thermal boundary layer is developed at the inlet of the mould (left side) because of the difference in temperature between the injected resin and the walls of the mould. Normally, at the flow front, there should be a sudden drop of temperature from 400 K to 340 K. However, because of numerical diffusion, the thermal front has stretched a little. This is what we would like to improve with remeshing algorithms.

5.5.2.2 Simulation with remeshing

The same parameters as previously for the fixed mesh are used in this simulation. As shown in [34] for Darcy problem, the number of time steps can be strongly decreased. It has been set to 50, thus the average time step is about 5 *s*. There are 4 layers of element in the front. Each time step includes one calculation of Darcy's flow (Eq. (61)), followed by one calculation of the thermal field as described above (Eq (76)).

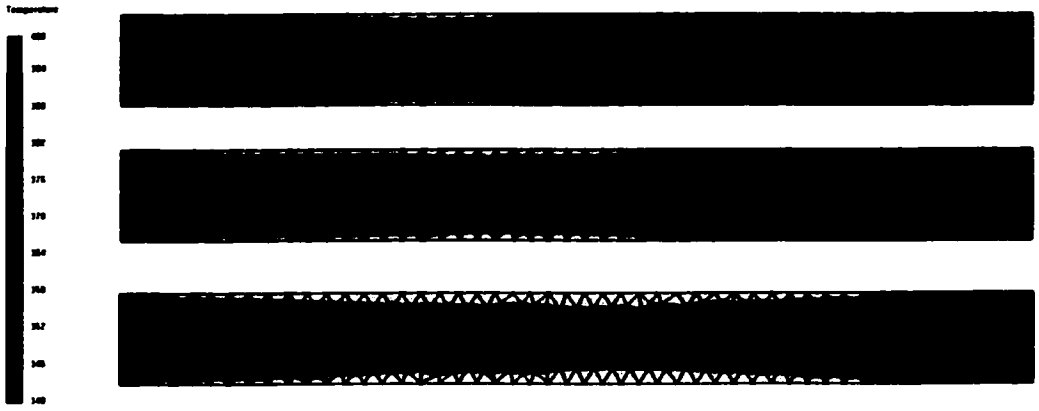


Figure 5.6: Temperature calculated with an adaptive mesh at 50*s*, 100*s* and 200*s* after the beginning of injection

The results shown in Figure 5.6 are not satisfactory. The thermal boundary layer is not stable, and the temperature in the mould drops far below 340 *K* in some places and rises slightly above 400 *K* elsewhere, which are the physical limits in the problem considered here. This observed numerical instability is connected with conditions on the Courant number that will be discussed in the sequel. Let us return to the fixed mesh simulation. The only differences between this simulation and the current one on an adaptive mesh are the time step, and the re-interpolation that occurs at each time step.

5.5.2.3 Results with a fixed mesh for fewer (larger) time steps

The next simulation is performed by forcing the filling algorithm to use longer time steps. The number of time steps is around 45. The duration of each time step is 7*s*. This value is close to the value used in the moving mesh simulation. The only difference is that the mesh is now fixed. In that specific case, there is no numerical diffusion added by mesh re-interpolation between each time step.

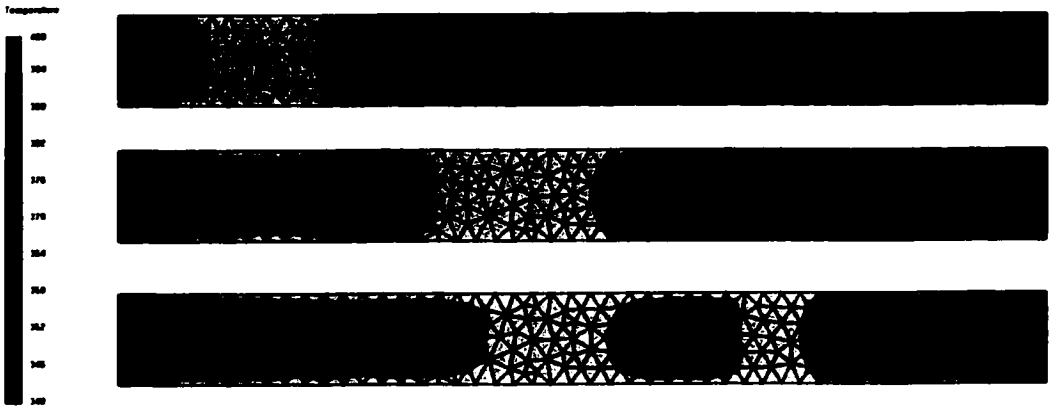


Figure 5.7: Temperature calculated with longer time steps on a fixed mesh at 50*s*, 100*s* and 200*s* after the beginning of injection

As one can see in Figure 5.7, similar features as in the remeshing simulation appear also in that case. The thermal boundary layer is not developed. In other regions of the mould, the temperature exceeds the physical limit of 400 K and drops below 340 K. A new "feature" is noticeable here, namely the presence of "bubbles" of different temperature carried along by the flow. These oscillations in the temperature field are due to the value of the time step used for the simulation. The local Courant number is defined by :

$$C_g = \frac{\|\mathbf{v}\| \cdot \Delta t}{\Delta x} \tag{77}$$

where Δx is the characteristic length of the element. In general, this length is evaluated by:

$$\Delta x = \sqrt[d]{V} \tag{78}$$

where d is an integer denoting the dimension of the element; V is the volume of the element (its surface for 2D elements, and length for 1D ones), and Δt is the time step.

Courant number is the ratio between the time step, and the time needed for the flow to come across one layer of elements. In the above simulation as well as in the one involving remeshing, $C_g = 1.5$. This mean that the "information" on the thermal field is transmitted across more than one new element at a time at each calculation step. For example, if the numerical scheme was a centered finite difference, the actual stability requirement would be $C_g \leq 0.5$, among other requirements over the Peclet grid number Pe_g for example which is defined by :

$$Pe_g = \frac{\|\mathbf{v}\| \cdot \Delta x}{\alpha} \tag{79}$$

and the factor α, namely the thermal diffusivity defined by :

$$\alpha = \frac{\mathbf{v} \cdot \mathbf{k} \cdot \mathbf{v}}{\|\mathbf{v}\| \cdot \rho \cdot c_p} \tag{80}$$

The Peclet grid number is not involved as the numerical scheme used here [48] can deal with purely convective flows (which is not the case for standard Galerkin formulations). Moreover, Peclet number does not depend on the time step, which is the only parameter to vary from the parameters of the previous simulations. In the first simulation, the Courant number was around 0.25.

5.5.2.4 Modification of the remeshing algorithm

To cope with the problems arising with the longer time steps used in the previous examples, shorter time steps were used in the thermal problem. The time step was

reduced in order to obtain a Courant number lower or equal to $C_g^{max} = 0.5$. This means that the maximum time step must be lower than :

$$\Delta t_{max} = \frac{C_g^{max} \cdot \Delta x}{\|\mathbf{v}\|} \tag{81}$$

Depending on the refinement in the vicinity of the flow front, this leads to a time step 4 to 5 times shorter than the value used in the second simulation. Thus, for each resolution of Darcy's problem, giving the velocity v and updating the front position, four sub-time steps are necessary to solve the thermal problem. Figure 5.8 shows no instability, despite the small perturbation at the beginning of the injection. In addition, there is a relatively narrow thermal boundary layer in the vicinity of the flow front when compared to the fixed mesh simulation. This was a kind of result expected from a remeshing approach.

However, one of the initial objectives of remeshing, namely to reduce computational effort [34], is not fully met. It is clear that in a thermal mould filling simulation, most of the computer time is used to solve the convection-diffusion heat exchange problems.

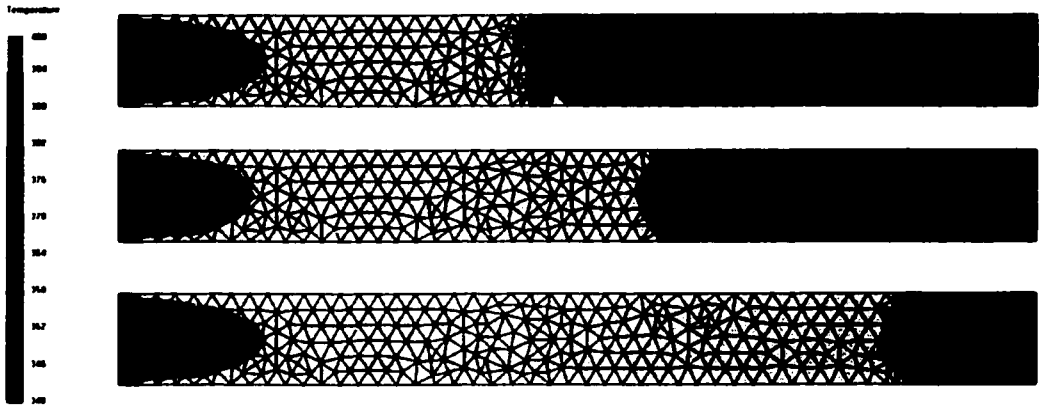


Figure 5.8: Temperature calculated with longer time steps (and 5 sub-time steps for thermal analysis) at 50*s*, 100*s* and 200*s* after the beginning of injection

5.6 Adapted mesh for radial injection

Figure 5.9 shows a radial injection configuration where R_i is the radius of the injection port, and R_e is the external radius of the mould. The injection pressure is P_i, and the vent pressure is set to zero. At time t the front is at position r. For symmetry reasons, the solution is radial and only half of this part needs to be simulated. In this example, the analytical solution is known. First, we shall recall the filling equations for this simple geometry. Then an analysis that covers four ways to generate a mesh is presented.

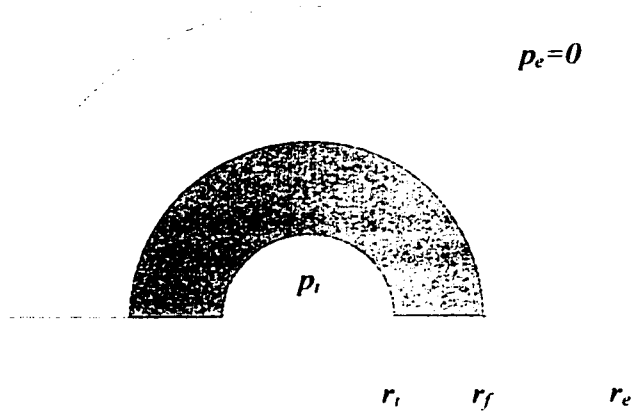


Figure 5.9: Central injection at constant pressure. The grey portion represents the fluid saturated region at time t

5.6.1 Analytical solution of Darcy's equation in radial coordinates

The front is moving towards the periphery of the mould. An analytical solution will be determined for comparison with numerical results. Darcy's in radial coordinates writes as follows in the saturated domain:

$$\frac{dr}{dt} = -\frac{K}{\mu\phi}\cdot\frac{dp}{dr} \tag{82}$$

Along with the condition of mass conservation, assuming K constant in the saturated domain yields a radial differential equation :

$$\frac{1}{r}\frac{dp}{dr} + \frac{d^2p}{dr^2} = 0 \tag{83}$$

Solving this equation yields the well known logarithmic expression of the pressure field :

$$p(r) = p_i \left(1 - \frac{\ln \frac{r}{r_i}}{\ln \frac{rf}{ri}} \right) \tag{84}$$

From equation (84) we have

$$\frac{dp}{dr} = -\frac{P_i}{r \ln \frac{r_f}{r_i}} \tag{85}$$

Using Darcy's law, i.e. equation (82), this gives the velocity of the resin inside the saturated domain :

$$\frac{dr}{dt} = \frac{K}{\mu \phi} \cdot \frac{P_i}{r \ln \frac{r_f}{r_i}} \tag{86}$$

Thus, at the flow front, $r = r_f$:

$$\frac{dr_f}{dt} = \frac{K}{\mu \phi} \cdot \frac{P_i}{r_f \ln \frac{r_f}{r_i}} \tag{87}$$

Integration of this differential equation can be carried out to get $t(r_f)$. Separation of variables gives

$$\int r_f \ln\frac{r_f}{r_i} dr_f = \int \frac{Kp_i}{\mu\phi} dt + C_1 \tag{88}$$

which leads to

$$t(r_f) = \frac{\mu\phi}{4Kp_i} r_f^{\,2} \cdot \left(2\ln\frac{r_f}{r_i}\right) + C_2 \;;\; t(r_i) = 0 \tag{89}$$

Finally,

$$t(r_f) = \frac{\mu\phi}{4Kp_i}\left(r_f^{\,2} \cdot \left(2\ln\frac{r_f}{r_i} - 1\right) + r_i^{\,2}\right) \tag{90}$$

5.6.2 Mesh adaptation for a radial injection

Assuming that the filling algorithm progresses one layer of elements between each simulation loop, the relation between the time step Δt and the size of the elements Δr near the flow front ($r = r_f$) is $\Delta r = \frac{dr}{dt}\Delta t$.

Now we shall determine a size map suitable for mesh generation. In this radial problem, the tangential size of the elements, namely S_θ, can be chosen arbitrarily. In the sequel, it will always be "as coarse as possible", with respect to the curvature of the front. The radial size S_r, however, depends on the use of the mesh. Four equally valid mesh

adaptation strategies will be presented. Two of them refer to "user friendly" parameters, namely a constant time step (for an isochrone mesh), and a constant radial mesh size (isoparametric mesh). Both the other two strategies refer to numerical errors. When a transport equation is involved, one can desire a mesh which levels the Courant number (iso-Courant mesh). Or, one can choose to level the interpolation error on the pressure variable p. This leads to a different mesh (iso-error mesh). The aim of this study is to show that one can generate many different meshes with a single thermal injection problem, and those mesh requirements are not necessarily compatible.

5.6.2.1 Isochrone mesh

In this case, we expect the time step to remain constant. Thus, a simulation conducted with this mesh will show results at regularly spaced times. Equation (87) gives for $r_f=r$:

$$\frac{dr}{dt} = \frac{Kp_i}{\mu\phi r \ln\frac{r}{r_i}} \tag{91}$$

A first order approximation here gives the following radial size mesh for an isochrone adaptation:

$$S_r^{chron}(r) = \Delta r = \frac{Kp_i}{\mu\phi r \ln\frac{r}{r_i}} \cdot \Delta t \tag{92}$$

5.6.2.2 Isoparametric mesh

In this case, the mesh is expected to be have a constant radial size. For consistence with the isochrone mesh and in order to make comparisons possible, the actual parameter

describing the size map will still be the time step Δt. This time the ratio $\frac{\Delta r}{\Delta t}$ will be evaluated at an average position between r_i and r_e in the circular mould: $r = r_m = \frac{r_i + r_e}{2}$. This convention will be used also in the further cases studied of the iso-Courant and iso-error meshes. At $r_f = r_m$, equation (87) gives:

$$\frac{dr}{dt} = \frac{Kp_i}{\mu \phi r_m \ln \frac{r_m}{r_i}} \tag{93}$$

so the size map for an isoparametric mesh is:

$$S_r^{param}(r) = \Delta r = \frac{Kp_i}{\mu \phi r_m \ln \frac{r_m}{r_i}} \cdot \Delta t \tag{94}$$

5.6.2.3 Iso-Courant mesh

In this case, the adapted mesh is expected to provide a constant Courant number when a transport equation is solved. The aim here is to help stabilize the numerical scheme used to solve the transport equation and to minimize numerical diffusion. The filling algorithm is supposed again to fill one layer of elements at each time step. The definition of the Courant number is taken from equation (77):

$$C_g(r,t) = \frac{\frac{dr}{dt} \cdot \Delta t(t)}{\Delta r(r)} = C \tag{95}$$

Again, the time increment Δt is set for $r_f = r_m$, the average radial positon in the mould. Equation (86) gives then for any r:

$$\frac{dr}{dt} = \frac{Kp_i}{\mu\phi r \ln \frac{r_m}{r_i}} \tag{96}$$

Along with the constant Courant number, this yields:

$$\Delta r = \frac{Kp_i}{\mu\phi r \ln \frac{r_m}{r_i}} \cdot \frac{\Delta t}{C} \tag{97}$$

If one layer of element is filled at each time step, obviously $C = 1$ near the front. Since this parameter must remain constant, the size map for an iso-Courant mesh is:

$$S_r^{courant}(r) = \Delta r = \frac{Kp_i}{\mu\phi r \ln \frac{r_m}{r_i}} \cdot \Delta t \tag{98}$$

It should be mentioned that this result is valid at any time and anywhere in the domain. In addition, it is possible to make a simulation while keeping an almost constant Courant number, provided that 1) the physical properties are uniform, 2) the flow front remains unique and 3) there are no discontinuities in the boundaries of the mould. In some specific cases, as the one exposed here, the courant number can be strictly kept constant.

5.6.2.4 Iso-error mesh

In this case, we expect the mesh to provide a constant interpolation error for the pressure field. Of course, because the mesh is fixed and the pressure field changes much, it is impossible to ensure a constant interpolation error in time in the case of a pressure driven injection. Again, we shall set the time increment Δt for $r_f = r_m$. This gives Δr at that position and the relative error that must remain constant (when the cavity is completely injected) :

$$\left.\frac{d^2p}{dr^2}\right|_{\substack{r_f=r_c \\ r=r_m}} \cdot \left.\frac{(\Delta r)^2}{p_i}\right|_{r=r_m} = E \tag{99}$$

From equation (84), we have :

$$\frac{d^2p}{dr^2} = \frac{p_i}{r^2 \ln \frac{r_f}{r_i}} \tag{100}$$

So the relative error is:

$$E = \frac{K^2 p_i^{\,2}}{\mu^2 \phi^2} \cdot \frac{\Delta t^2}{r_m^{\,4} \left(\ln \frac{r_m}{r_i} \right)^3} \tag{101}$$

By setting this value to be the error at any radius r we get:

$$\frac{d^2p}{dr^2}\cdot\frac{(\Delta r)^2}{p_i}=E \tag{102}$$

, so

$$(\Delta r)^2=\frac{K^2p_i^{\,2}}{\mu^2\phi^2}\cdot\frac{r^2(\Delta t)^2}{r_m^{\,4}\left(\ln\frac{r_m}{r_i}\right)^2} \tag{103}$$

Finally, the size map for an iso-error mesh is

$$S_r^{error}(r)=\Delta r=\frac{Kp_i}{\mu\phi}\cdot\frac{r\Delta t}{r_m^{\,2}\ln\frac{r_m}{r_i}} \tag{104}$$

5.6.3 Numerical results and discussion

The geometry selected in these tests has the following internal and external radii: $r_i=2$. $r_e=10$. ($r_m=6$). The physical constants are such that $\frac{Kp_i}{\mu\phi}=1.0$. The results are shown here for $\Delta t=1.5\ s$. The tangential size is set to $S_\theta(r)=\frac{0.25r}{r_i}$. and the radial size $S_r(r)$ is derived from equations (92), (94), (98) and (104).

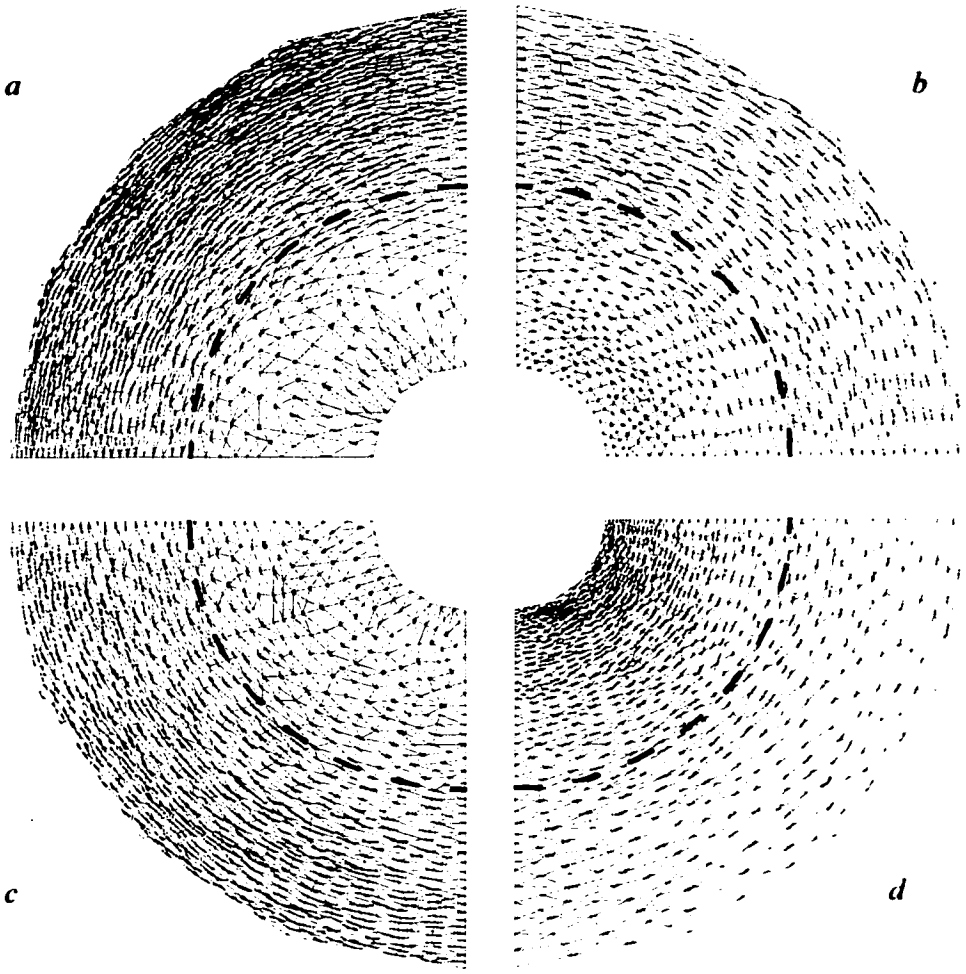


Figure 5.10: Meshes generated to fit particular purposes : (*a*) isochrone mesh (1802 elements); (*b*) isoparametric mesh (1264 elements); (*c*)- iso-Courant mesh (1386 elements) and (*d*)- iso-error mesh (1522 elements). The dashed line denotes the average of the cavity at which every mesh has the same density.

From Figure 5.10 it is clear that a mesh generated to lower the interpolation error, typically designed to optimise filling simulations such as mesh (*d*) is not appropriate to solve the heat transport equation in thermal problems. In that case, the time step is constrained by the CFL condition, and will be very small because of tiny elements near

the injection gate (where the velocity is the highest). For such a calculation, mesh (*c*) is the most adapted one. This sets some limits to the improvement in accuracy that can be expected from a mesh adaptation in a coupled filling and thermal analysis. Meshes (*d*) and (*c*) are clearly not compatible because one is generated with the smallest size satisfying a CFL condition, given a time step, while the other is generated as the coarsest mesh allowing a given interpolation error. On the other hand, meshes (*a*) and (*b*) are more "user friendly" and could be generated for the user's convenience more than for their numerical suitability. An isochrone mesh is not practically suitable for central injections because the error is too high at the beginning of the injection. The isoparametric mesh (*b*) represents a good compromise between an iso-Courant and an iso-error mesh, but is not optimal. There is no optimal mesh if one want to simulate non isothermal filling, instead one should have compromises.

5.7 Conclusion

This paper describes a non isotropic remeshing algorithm and its application to RTM simulations. First, we show the need for a specific error estimator to generate *a priori* adapted meshes. Then the remeshing algorithm was implemented to simulate non isothermal mould filling. The result obtained was not satisfactory at first because the numerical method used to solve the thermal problem was not able to consider large Courant numbers. The time step must be adapted to keep the Courant number sufficiently small, thus limiting the gain in computational cost that can be achieved with adaptive algorithms. However, the algorithm allows to decrease considerably the numerical diffusion near the flow front. This represents a definite improvement brought about by adaptive remeshing procedures in the simulation of convective heat transfer mould filling problems. Another positive outcome of this analysis is connected with the kind of mesh refinement that can be performed in the vicinity of a radial inlet port. Among the four cases considered, the iso-Courant and iso-error schemes provide very different mesh refinements. A reduction in the size of the elements is suitable to reduce

the approximation error on pressure in mould filling simulations. However, such a mesh is not suited to perform a convective thermal analysis. A compromise can be found by using an isoparametric mesh, i.e., a mesh of constant radial increment. Part II of this investigation concerns the application of adaptive remeshing to inject shell-type parts, that can be modelled by three-dimensional surfaces.

5.8 Acknowledgements

We acknowledge the Natural Science and Engineering Research Council of Canada (NSERC) and the Fonds Québécois de Recherche sur la Nature et les Technologies (FQRNT) for their financial support.

5.9 References

[32] D. Adalsteinsson, J.A. Sethian, 1999, The Fast Construction of Extension Velocities in Level Set Methods, J. of Computational Physics **148**, 2-22.

[33] E. Béchet, J.C. Cuillière, F. Trochu, 2002, Generation of a finite element mesh from stereolithography (STL) files, Computer Aided Design **34**, 1-17.

[34] E. Béchet, E. Ruiz, F. Trochu, J.C. Cuillière, 2003, Remeshing Algorithms Applied to Mould Filling Simulations in Resin Transfer Moulding, to appear in Journal of Reinforced Plastics and Composites .

[35] E. Bohr, J.F. Remacle, F. Trochu, 2002, Simulation of Heat Transfer and Curing in Resin Transfer Molding, to be submitted to Journal of Heat Transfer .

[36] M.V. Bruschke, S.G. Advani, 1990, A Finite Element/Control Volume Approach to Mold Filling in Anisotropic Porous Media, Polymer Composites, **11** (6), 398-405 .

[37] M.V. Bruschke, S.G. Advani, 1994. A Numerical Approach to Model Non-Isothermal Viscous Flow Through Fibrous Media With Free Surface, Int. J. Num. Meth. Fluids **19**, 575-603 .

[38] G.C. Buscaglia, E.A. Dari, 1997. Anisotropic mesh optimization and its application in adaptivity, Int. J. Num. Meth. Engng. **40** (22), 4119-4136 .

[39] D.R. Calhoun, S. Yalvaç, D.G. Wetters, C.H. Wu, T.J. Wang, J.T. Tsai, L.J. Lee, 1996. Mold Filling Analysis in Resin Transfer Molding, Polymer Composites **17** (2),251-264 .

[40] W. Chang, N. Kikuchi, 1994. An Adaptive Remeshing Method in Simulation of Resin Transfer Molding (RTM) Process, Computer Methods in Applied Mechanics and Engineering 112, 41-68 .

[41] Y.F. Chen, K.A. Stelson, V.R. Voller, 1997. Prediction of Filling Time and Vent Location for Resin Transfer Molds, J. of Composite Materials 31 (11), 1141-1161 .

[42] B. Delaunay, 1934. Sur la sphère vide, Bul. Acad. Sci. URSS, Class. Sci.Nat., 793-800 .

[43] P.L. George, H. Borouchaki, 1997. Triangulation de Delaunay et maillages : application aux éléments finis, Paris, Hermès, ISBN 2-86601-625-4 .

[44] G. Guyonvarch, M. Audet, Y.-Y. Qian, F. Trochu, D. Delaunay, 1996, Validation of Non Isothermal Resin Transfer Molding Simulations, Composites **14**, 101-106 .

[45] R. Kimmel, J.A. Sethian, 1998, Fast Marching Methods on Triangulated Domains, Proc. Nat. Acad. Sci. **95**, 8341-8435 .

[46] R. S. Maier, T. F. Rohaly, S. G. Advani, K. D. Fickie, 1996, A Fast Numerical Method for Isothermal Resin Transfer Mold Filling, International Journal of Numerical Methods in Engineering **39**, 1405-1422 .

[47] C.C. Pain, A.P. Umpleby, C.R.E de Oliveira, A.J.H. Goddard, 2001, Terahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations, Comp. Meth. Appl. Mech. Eng. **190**, 3771-3796 .

[48] L. Lesaint, P. Raviart, 1974, On a Finite Element Method for Solving the Neutron Transport Equation, in Mathematical Aspects of Finite Elements in Partial Differential Equations, Carl de Boor ed., Academic Press, New-York .

[49] R. J. Lin, L. J. Lee, M. J. Liou, 1993, Mold filling and curing analysis liquid composite molding, Polymer Composites **14** (1),71-81.

[50] F. Muttin, T. Coupez, M. Bellet, J.L. Chenot, 1993, Lagrangian Finite Element Analysis of Time-Dependent Viscous Free-Surface Flow Using an Automatic Remeshing Thechnique : Application to Metal Casting Flow, Inter. J. Numer. Meth. Eng. **36**, 2001-2015.

[51] J. Ruppert, 1994, A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation, Journal of Algorithms **18**, 548-585 .

[52] J.A. Sethian, 1996, Fast Marching Level Set Method for Monotonically Advancing Fronts, , Proc. Nat. Acad. Sci. **93** (4), 1591-1595 .

[53] J.A. Sethian, 1999, Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision and Materials Science, Cambridge University Press .

[54] F. Trochu, M. Audet, R. Gauvin, Y. Beguin, 1997, Non-Isothermal Analysis of Mold Filling in Liquid Composite Molding, Fifth International Conference on Automated Composites (ICAC-97), Glasgow, U.K., September 4-5, 1997 .

[55] F. Trochu, P. Ferland, R. Gauvin, 1997, Functional Requirements of a Simulation Software for Liquid Molding Processes, Science & Engineering of Composite Materials **6** (4), 209-218 .

[56] F. Trochu, R. Gauvin, D.M. Gao, 1993, Numerical Analysis of the Resin Transfer Molding Process by the Finite Element Method. Advances in Polymer Technology **12** (4), 329-342 .

[57] C.L. Tucker, R.B. Dessenberger, 1994, Governing equations for flow and heat transfer in stationnary fiber beds, In S.G. Advani : Flow and Rheology in Polymer Composites Manufacturing, Elsevier, Amsterdam. 257-323 .

[58] C.L. Tucker, 1996, Heat Transfer and Reaction Issues in Liquid Composite Molding, Polymer Composites **17** (1), 60-72 .

[59] W.B. Young, K. Rupel, K. Han, L.J. Lee, M.J. Liou, 1990, Simulation and Experimental Verification of Mold Filling in Resin Transfer Molding and Structural RIM, 45th Annual Conference, Composite Institute, The Society of Plastics Industry .

[60] W.B. Young, K. Han, L.H. Fong, L.J. Lee, 1991,Flow Simulations in Moulds with Preplaced Fibre Mats, Polymer Composites **12** (6), 391-403 .

[61] A. Zaki, Simulation numérique des problèmes de convection sur des maillages adaptatifs non structuré du type h-p, 1993, Phd thesis, Department of Mathematics, Ecole Polytechnique de Montréal .

CHAPITRE 6

ADAPTIVE MESH GENERATION TO SOLVE MOULD FILLING PROBLEMS WITH APPLICATION TO RESIN TRANSFER MOULDING - PART II: CASE OF THREE-DIMENSIONAL SURFACES

E. BÉCHET[1,2], J.-C. CUILLIERE[2], F. TROCHU[1]

(1) Centre de Recherches Appliquées Sur les Polymères (CRASP), Département de Génie Mécanique, École Polytechnique de Montréal, H3C 3A7, Canada
eric.bechet@epost.de ; francois.trochu@polymtl.ca

(2) LIRICS,, Département de Génie Mécanique, Université du Québec à Trois-Rivières, G9A 5H7, Canada
jean-christophe_cuilliere@uqtr.ca

6.1 Abstract

Adaptive algorithms for two-dimensional anisotropic mesh generation have been described in Part I of this contribution. The purpose was to lower the computational cost of RTM (Resin Transfer Moulding) simulations and improve the accuracy of mould filling calculations by implementing an *a-priori* adapted mesh generation algorithm. In this part, these ideas are generalized to three dimensional shells, which represent the most common type of composite parts manufactured by RTM. In particular, an error estimator generally used in planar or volumetric geometries is extended to curved surfaces. The extension consists of a projection of the solution field in the tangent plane to avoid problems related to the locally curved geometry of the part. In order to illustrate these concepts, the adaptive algorithm is implemented in the case of RTM flow simulations. Finally, a simulation is carried out on a real part.

Keywords: Resin transfer moulding, finite elements, Darcy equation, anisotropic mesh generation, mesh adaptation, error estimator, surface mesh generation.

6.2 Introduction

This paper proposes a new way to improve at the same time the accuracy and efficiency of mould filling simulations. The adaptive algorithm is used to generate a pre-optimised mesh that will decrease the numerical errors by taking into account the geometry of the part and the material properties of the reinforcement. Part I of this article was focused on the two-dimensional case. This work is now extended for curved surfaces.

Most of the industrial parts manufactured by Resin Transfer Moulding (RTM) are three-dimensional thin shells. Since the thickness of those parts is usually small compared to the other dimensions, a surface mesh can be used to perform complex mould filling simulations as shown by Bruschke and Advani [66]. When a fixed mesh is involved as in most simulation methodologies developed in the last decade (Bruschke and Advani [65], Youg et al. [74] and Trochu et al.[71]), no specific issues related to surface mesh generation are involved apart from the generation of the mesh itself. This is usually done in a CAD system prior to the simulation. This leads generally to an isotropic mesh with does not take into account the particular features of the numerical problem that is solved. When mesh adaptation is performed as proposed by Béchet et al. [64], special care is needed in order to deal with curved surfaces.

Error estimators in the finite element method have been widely used in many engineering fields, including structural analysis (Lagrangian formulation), computational fluid dynamics (CFD) (Eulerian formulation), electromagnetism, among many others. A comprehensive overview can be found in Ainsworth and Oden [62], as well as in Verfürth [73]. Most of the error estimators (eg. Zienkiewicz and Zhu, [75] and [76]) actually include the geometrical error and the functional error "all in one". This is desirable when the exact geometry is known, because the mesh adaptation will bound the overall approximation error, no matter if it is of functional or geometrical origin. However, one issue arises when the geometry is not smooth. One has to provide special

treatment of discontinuities in order to avoid useless increase in mesh density in their vicinity. The typical example is a flow in a folded rectangular shape, as illustrated in Figure 6.1. The exact solution is linear along the lateral edge, for example. The error induced by the linear finite element approximation is zero, provided that the fold is part belongs to the discrete model (i.e., it is located along the edges of finite elements). This is always the case, since both patches located on either sides of the fold are meshed independently, and then joined in the CAD system.

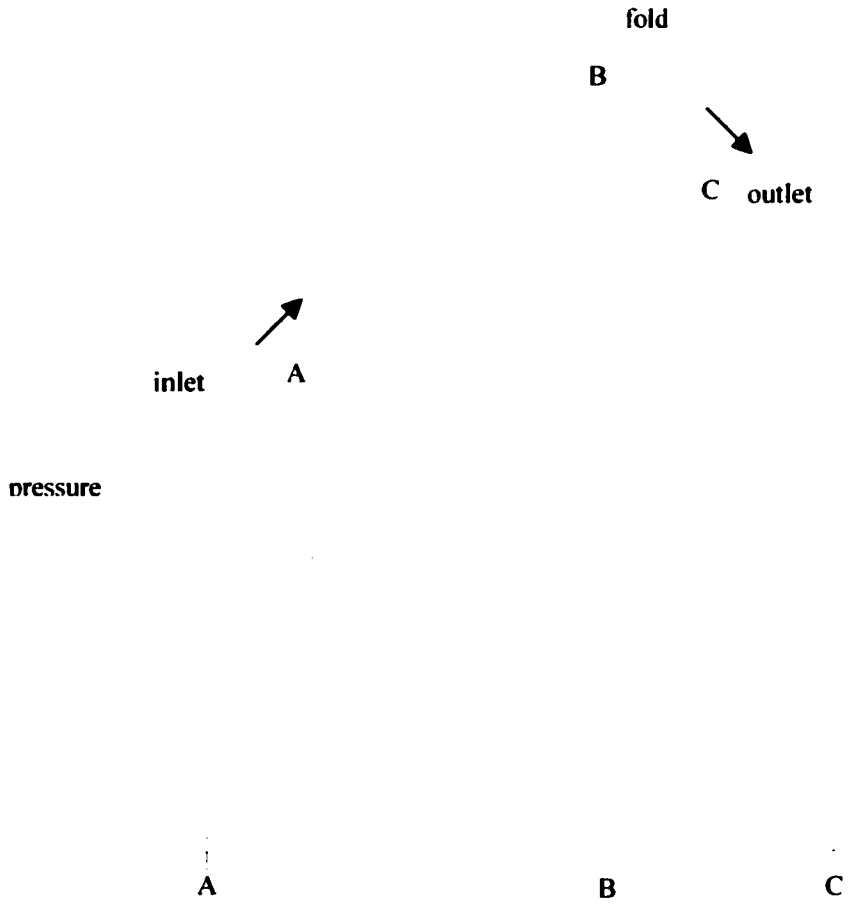


Figure 6.1: Illustration for a folded plate. The pressure is linear along the shape (arbitrary units)

However, the error indicated by a common error estimator won't vanish in the vicinity of the fold, because the gradient of the controlled variable (pressure in this case) is not continuous. The origin of this discontinuity is not functional, but rather purely geometrical. Because the finite element approximation can handle such geometrical discontinuities, there is no need of mesh refinement. The error estimator presented in the sequel will address this issue.

Particular aspects of surface mesh generation will be briefly described. Then, new results on non-isotropic error estimation will be presented, as well as an extension in the particular case of discrete curved surfaces. The error estimator presented in the sequel will allow to use a discrete geometry to perform adaptive mesh generation in the case of RTM simulations. Finally, an industrial part will be tested to illustrate this approach.

6.3 Surface mesh generation

An initial mesh must be provided, that must be fine enough to represent the geometry of the part, typically a STL file. The only requirement of this mesh is the conformity. Elements need not be of any prescribed size, except for geometry and boundary condition to be well represented. A standard bisection algorithm is used to refine the original mesh. This methodology was described in Béchet et al. [63] for the isotropic scheme. An extension has been made since for anisotropic mesh generation [64]. It is briefly summarized in the part I of this article. When dealing with 3-dimensional surfaces, however, some modifications had to be done to the mesh generation scheme. The new nodes have to be projected on the original surface that serves as a geometrical basis, to preserve the mesh from degenerating and loosing much of its geometric information. The algorithm used here is based on a proximity search to avoid to check the whole surface for the locus of the projected node. Also, the conventional Delaunay criterion must be adapted here to the discrete representation of a curved surface with the

initial background mesh [63][68]. Figure 6.2 shows example of surface meshes generated with those algorithms.

Figure 6.2: A Moebius band with an isotropic mesh (top) and an anisotropic mesh (bottom)

6.4 Error estimation for discrete surfaces

6.4.1 Anisotropic error estimator and metric

An error estimator has been described in Part I of this article for the 2-dimensional case. We shall recall briefly the results here.

The error estimator considered here is based on the second derivatives of the primal variable of the finite element formulation (pressure p for a Darcy flow, temperature T for a heat transfer problem, etc..). The Hessian matrix that will act as error estimator is defined as follows in a tree-dimensional space:

$$\mathbf{H}(p)=\begin{bmatrix} \frac{\partial^2 p}{\partial x^2} & \frac{\partial^2 p}{\partial y \partial x} & \frac{\partial^2 p}{\partial z \partial x} \\ \frac{\partial^2 p}{\partial x \partial y} & \frac{\partial^2 p}{\partial y^2} & \frac{\partial^2 p}{\partial z \partial y} \\ \frac{\partial^2 p}{\partial x \partial z} & \frac{\partial^2 p}{\partial y \partial z} & \frac{\partial^2 p}{\partial z^2} \end{bmatrix} \tag{105}$$

This matrix is symmetrical for a continuous pressure field. From the Hessian matrix, it is easy to define a related metric as described in Part I. We use the eigenvalues and eigenvectors $\{\mathbf{v}_i\}$ of the Hessian matrix, which becomes then diagonal :

$$\mathbf{\Lambda}=\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \tag{106}$$

We denote $\mathbf{E}=\{\mathbf{v}_i\}$ the eigenvectors such that :

$$\mathbf{H}={}^{T}\mathbf{E}\cdot\mathbf{\Lambda}\cdot\mathbf{E} \tag{107}$$

By replacing the eigenvalues by their absolute values, we have :

$$|\mathbf{\Lambda}|=\begin{bmatrix} |\lambda_1| & 0 & 0 \\ 0 & |\lambda_2| & 0 \\ 0 & 0 & |\lambda_3| \end{bmatrix} \tag{108}$$

The following metric can be defined :

$$\mathbf{M}(p) = {}^{T}\mathbf{E} \cdot |\Lambda| \cdot \mathbf{E} \cdot \frac{1}{\varepsilon} \tag{109}$$

This metric will drive the mesh generation algorithm to produce a mesh that will bound the absolute interpolation error by a constant ε over the whole computational domain.

6.4.2 Evaluation of the Hessian matrix for linear finite elements

The evaluation of $\frac{\partial^2 p}{\partial x_i \partial x_j}$ cannot be made separately over each element as shown in Part I. Thus, the relation is used to get the 1st derivatives at the nodes of the considered element :

$$\left.\frac{\partial p}{\partial x_i}\right|_k \approx q_{x,k} = \frac{\sum_l V_l \left.\frac{\partial p}{\partial x_i}\right|_{E_l}}{\sum V_l} \tag{110}$$

It is then easy to calculate a piecewise constant approximation for each element E_o of the mesh with the help of the inverse $\mathbf{J}^{-1}$ of the Jacobian matrix used to change coordinates in the element considered:

$$\left.\frac{\partial^2 p}{\partial x_i \partial x_j}\right|_{E_o} = \mathbf{J}_{E_o}^{-1} \cdot \left.\frac{\partial\left(\frac{\partial p}{\partial x_i}\right)}{\partial u_k}\right|_{E_o} \tag{111}$$

$$\left.\frac{\partial\left(\frac{\partial p}{\partial x_i}\right)}{\partial u_k}\right|_{E_o} = \frac{\left.\frac{\partial p}{\partial x_i}\right|_k - \left.\frac{\partial p}{\partial x_i}\right|_l}{u_k|_k - u_k|_l} = \left.\frac{\partial p}{\partial x_i}\right|_k - \left.\frac{\partial p}{\partial x_i}\right|_l \tag{112}$$

for a linear interpolation over the element considered. Subscript "*l*" stands for the node defining the origin of the local coordinates in the element, and "*k*" denotes the other local nodes of the element. In the sequel, the Jacobian is augmented by the normalized vector product of the 2 base vectors (u, v) of the element. Of course, the normal component of the gradients to the element is considered (and determined) to be zero. For a triangle in the (x_1, x_2) plane, the Jacobian writes at follow :

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x_1}{\partial u_1} & \frac{\partial x_2}{\partial u_1} & 0 \\ \frac{\partial x_1}{\partial u_2} & \frac{\partial x_2}{\partial u_2} & 0 \\ 0 & 0 & 1 \text{ or } -1 \end{bmatrix} \tag{113}$$

6.4.3 Adaptation to a discrete surface geometry

The error estimator described above is standard for planar and solid geometries. However, if one uses this error estimator without modifications on a discrete surface geometry (i.e. a surface in $\Re^3$ made from triangular patches), it will lead to several artefacts as shown in Figure 6.3. This is due to the lack of smoothness of the underlying

geometry that supports the finite element space. This leads to singularities in the mesh if a minimal size is not set for the elements (the error is not scaled with the size of the elements, but remains roughly constant as the mesh is refined).

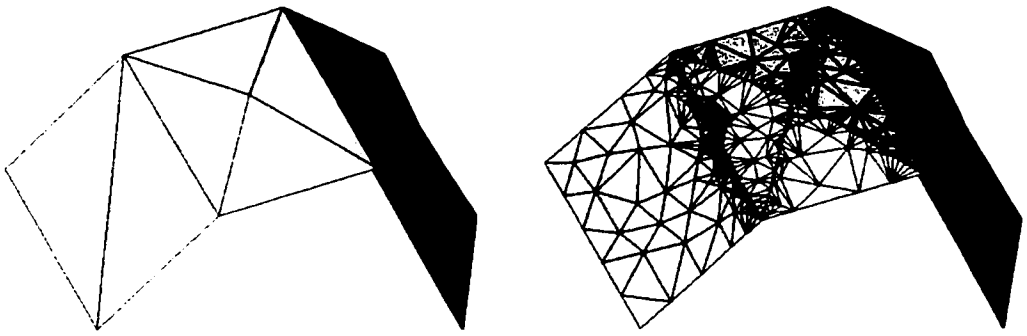


Figure 6.3: Original geometry and refined mesh for error adaptation (the isotropic case is shown here for more clarity)

The total error estimator $\mathbf{e}_T$ includes in fact two distinct errors :

$$\mathbf{e}_T = \mathbf{e}_G + \mathbf{e}_F \tag{114}$$

where $\mathbf{e}_G$ is the part of the error arising from the geometry because the triangular patches are not coplanar, and $\mathbf{e}_F$ is the part of the error coming from the interpolation itself.

There are two ways to avoid this problem. The first one is to construct a smooth C^1 geometry from the triangulated surface. This would require an interpolation with Nurbs [69] or other high order surface interpolation schemes such as kriging [70] for example. Information on the boundary conditions would need to be preserved. This means "creating" information from a model which is clearly an approximation of the real geometry (an STL file for example is not an exact representation for example, see [63]).

In order to avoid complex and possibly troublesome manipulations in order to define such a C^1 geometry, a second solution has been adopted. A new error estimator is constructed that is able to represent only the term $\mathbf{e}_F$ of the interpolation error. For this purpose the error estimator is calculated in a different way.

The gradient of the primary variable $\nabla p|_{E_n}$ is determined on every element $\mathbf{E}_n$ of the mesh. This is straightforward as the interpolation considered here is linear, thus ∇p is piecewise constant on each element of the domain :

$$\nabla p|_{E_n} = \mathbf{J}_{E_i}^{-1} \cdot \frac{\partial p}{\partial u_i} \tag{115}$$

$$\frac{\partial p}{\partial u_i} = \frac{p|_i - p|_l}{u_i|_i - u_i|_l} = p|_i - p|_l \tag{116}$$

As before, subscript "*l*" stands for the node defining the origin of the local coordinates, and "*k*" denotes the other local nodes of the element.

Let E_o denote the element considered here. For each of its neighbours E_k, $k = 1..3$, we will "rotate" the corresponding gradient vector $\nabla p|_{E_k}$ so that it lies in the plane containing element E_o (i.e., to make it fit into the vector subspace of element E_o). In Figure 6.4, $\mathbf{n}_o$ is the normal vector to element E_o, $\mathbf{n}_k$ is the normal vector to element E_k and $\boldsymbol{\delta}_k$ is the axis of rotation of the gradient for element E_k are determined as follows:

$$\boldsymbol{\delta}_k = \frac{\mathbf{n}_o \wedge \mathbf{n}_k}{\|\mathbf{n}_o \wedge \mathbf{n}_k\|} = \frac{\overrightarrow{AB}}{\|\overrightarrow{AB}\|} \tag{117}$$

The signed angle of rotation is determined by its sine and cosine:

$$\cos\theta_k = \frac{\mathbf{n}_o \cdot \mathbf{n}_k}{\|\mathbf{n}_o\| \cdot \|\mathbf{n}_k\|} \tag{118}$$

$$\sin\theta_k = \frac{\mathbf{n}_o \hat{} \mathbf{n}_k \cdot \boldsymbol{\delta}_k}{\|\mathbf{n}_o\| \cdot \|\mathbf{n}_k\|} \tag{119}$$

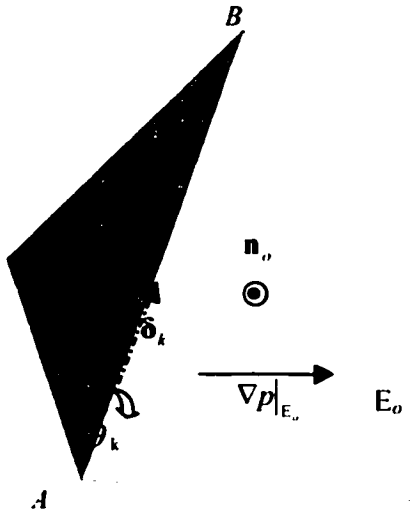


Figure 6.4: Original geometry and local gradient vectors of the primary variable (p)

Let $\mathbf{R}_k$ denote the rotation matrix of angle θ_k and axis $\boldsymbol{\delta}_k$. The new gradient $\nabla' p|_{E_k}$ in the plane of element E_o is then computed for each element E_k around E_o by the following formula :

$$\nabla' p|_{E_k} = \mathbf{R}_k \cdot \nabla p|_{E_k} \tag{120}$$

where

$$\mathbf{R}_k = \begin{bmatrix} \boldsymbol{\delta}_{k1}^2 + (1-\boldsymbol{\delta}_{k1}^2)\cos\theta_k & \boldsymbol{\delta}_{k1}\boldsymbol{\delta}_{k2}(1-\cos\theta_k) - \boldsymbol{\delta}_{k3}\sin\theta_k & \boldsymbol{\delta}_{k1}\boldsymbol{\delta}_{k3}(1-\cos\theta_k) + \boldsymbol{\delta}_{k2}\sin\theta_k \\ \boldsymbol{\delta}_{k1}\boldsymbol{\delta}_{k2}(1-\cos\theta_k) + \boldsymbol{\delta}_{k3}\sin\theta_k & \boldsymbol{\delta}_{k2}^2 + (1-\boldsymbol{\delta}_{k2}^2)\cos\theta_k & \boldsymbol{\delta}_{k2}\boldsymbol{\delta}_{k3}(1-\cos\theta_k) - \boldsymbol{\delta}_{k1}\sin\theta_k \\ \boldsymbol{\delta}_{k1}\boldsymbol{\delta}_{k3}(1-\cos\theta_k) - \boldsymbol{\delta}_{k2}\sin\theta_k & \boldsymbol{\delta}_{k2}\boldsymbol{\delta}_{k3}(1-\cos\theta_k) + \boldsymbol{\delta}_{k1}\sin\theta_k & \boldsymbol{\delta}_{k3}^2 + (1-\boldsymbol{\delta}_{k3}^2)\cos\theta_k \end{bmatrix} \quad (121)$$

The Hessian matrix can now be calculated from the new values of the gradient for each neighbouring element of E_o. In particular the gradients being known on both sides of each edge of the element E_o, the next few steps will allow to calculate the Hessian matrix:

1. The jump $\Delta_k(\nabla p)$ of the gradient trough each of the edges is defined as :

$$\Delta_k(\nabla p) = \nabla^r p\big|_{E_i} - \nabla p\big|_{E_o} \quad (122)$$

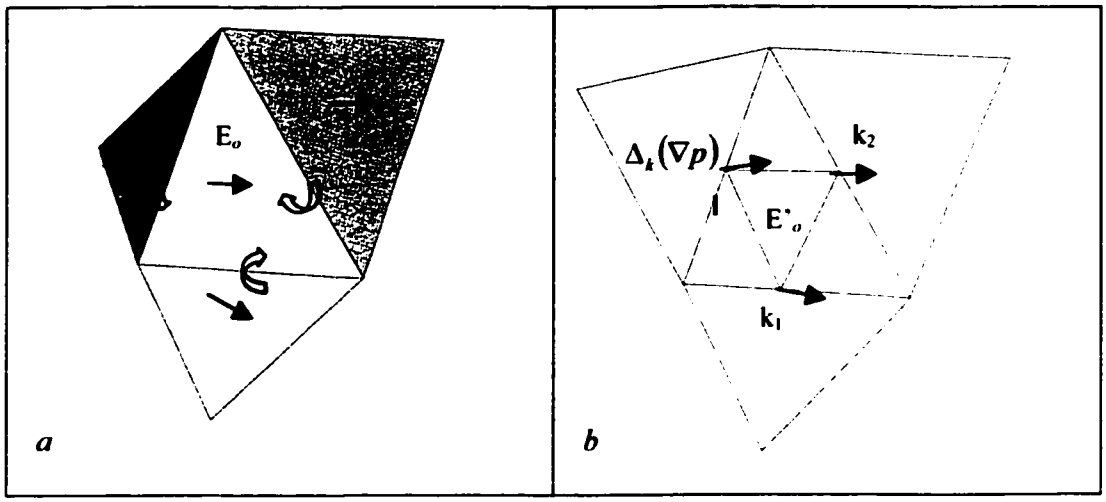


Figure 6.5: Original and locally flattened geometry. The arrows denote : *(a)* the gradient of the primary variable *p*; *(b)* its jump set at the middle of each edge of element E_o.

2. The jump is assigned to the midpoint of each edge *k* of element E_o. A new element E'_o is defined from those midpoints. On this element, we can interpolate the jump $\Delta_k(\nabla p)$ as a linear function of space.

3. The differentiation of this approximation of the gradient in element E'_o gives a constant 3-by-3 matrix. (Differentiation over each variable for each component of a vector leads to 9 independent second derivatives)

4. This approximation is extended from element E'_o back to the whole element E_o.

$$\left.\frac{\partial^2 p}{\partial x_i \partial x_j}\right|_{E_m} = \mathbf{J}^{-1}_{E_m} \cdot \left.\frac{\partial\left(\frac{\partial p}{\partial x_i}\right)}{\partial u_k}\right|_{E_m} \tag{123}$$

$$\left.\frac{\partial\left(\frac{\partial p}{\partial x_i}\right)}{\partial u_k}\right|_{E_m} = \frac{\Delta_k(\nabla p)_i - \Delta_l(\nabla p)_i}{u_k|_k - u_k|_l} = \Delta_k(\nabla p)_i - \Delta_l(\nabla p)_i \tag{124}$$

Again here, subscript "*l*" stands for the node defining the origin of the local coordinates, and "*k*" denotes the other local nodes of the element. In the triangular shell elements considered here, *k*=1 and 2 as illustrated previously in Figure 6.5.

5. Finally, the Hessian matrix is simply :

$$\mathbf{H}(p)|_{E_m} = \left.\frac{\partial^2 p}{\partial x_i \partial x_j}\right|_{E_m} = \left.\frac{\partial^2 p}{\partial x_i \partial x_j}\right|_{E_m} \tag{125}$$

The above Hessian matrix is generally not symmetric, because of the successive approximations from the original scalar function *p*. This approach leads also to an approximation that does not satisfy the following condition:

$$\nabla\wedge(\nabla p) = \vec{0} \tag{126}$$

In fact, the information needed in the computation (one central element plus up to 3 neighbours) makes it more prone to deviate from the nature of the "real" second

derivatives of a continuous scalar function. If needed, one can take the symmetrical part of the approximate Hessian matrix to be consistent with equation (126) as done in [67] :

$$\mathbf{H}(p)\Big|_{E_m}^{sym} = \frac{{}^{T}\mathbf{H}(p)\Big|_{E_m} + \mathbf{H}(p)\Big|_{E_m}}{2} \tag{127}$$

6.4.4 Results obtained with this error estimator

For a problem which admits a linear solution in pressure (for example, a solution of Darcy flow in a porous medium of constant section and shape), the finite element estimator should give a zero error everywhere because a linear finite element approximation is able to represent the exact solution. In this case, the error estimator has no influence. The correction of the estimator to account for the non planar geometry of the mesh yields now a uniform mesh. The remeshing algorithm generated a constant sized mesh (as prescribed) to avoid too large elements, which was not the case in the example of Figure 6.3.

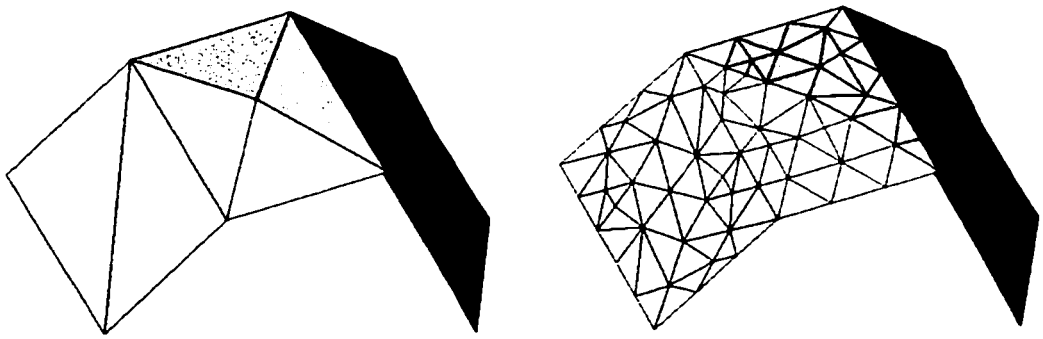


Figure 6.6: Original geometry and refined mesh for error adaptation with a corrected error estimator (isotropic case)

6.4.5 Application to mould filling simulations

After having defined the error estimator for curved surfaces, the adaptative finite element procedure used to solve mould filling problems is the same as in Part I :

iv) A fixed mesh is used for the whole simulation.

v) At each time step, a solution of the flow problem is found.

vi) This solution is used to update the filled region and advance the flow front for the next iteration.

We shell recall that the size of the elements determines in fact the duration of the time step due to the filling algorithm used here [72].

6.4.5.1 Application to Darcy's law

In shells, Darcy's law governs flow in the porous media the same way it does for two-dimensional cavities. The only difference is that the permeability tensor of the porous media might depend much on the geometrical properties of the shape, when draping is involved for example, and show more anisotropical aspects.

6.4.5.2 Adaptive algorithm

Adaptation of the mesh is made using a convergent adaptation loop as in Part I. This is done by calculating the solution of the flow in a completely filled cavity as many times as necessary, until convergence and low error.

The error considered here is based on the norm of the modified Hessian matrix of the pressure field. This norm defines an error estimator suitable for linear finite elements.

For arbitrary curved surfaces, equation (127) is used to define the metric for the pressure field p:

$$\mathbf{M}(p) = \frac{1}{\varepsilon} \cdot \left| \mathbf{H}(p)\right|_{E_w}^{sym} \right| \tag{128}$$

Recall here that the notation $\left|\mathbf{H}(p)\right|_{E_w}^{sym}\right|$ refers to the same matrix, but with the absolute value of its eigenvalues as defined in equation (109). This metric is generally anisotropic. If the reinforcement is isotropic, there is no need for anisotropic mesh generation as the diffusion of the fluid in the cavity is generally isotropic. For that purpose, the following metric will be used:

$$\mathbf{M}_{\varepsilon}(p) = \frac{1}{\varepsilon} \cdot \mathbf{I} \cdot \max_i(|\lambda_i|) \tag{129}$$

In the above equation, λ_i are the eigenvalues of $\mathbf{H}(p)\big|_{E_w}^{sym}$, and $\mathbf{I}$ is the identity matrix.

The user provides an upper bound ε of the error, which controls the adaptation algorithm in order to generate a mesh that will guarantee a given accuracy in the filling simulation.

As in part I, the problem involves also the convection of chemical species and heat when non isothermal injection conditions are set. The stretching/compression of elements along the flow directions may be useful to lower the numerical diffusion brought by various numerical schemes (Lesaint-Raviart in particular). Therefore, the metric defined above for a strictly diffusive problem must include some information regarding the hyperbolic aspect of the true physical problem. Besides the "elliptic"

metric of equation (128), a new "hyperbolic" metric, based on the local velocity **v** of the flow, is introduced below. This is valid in a global referential for curved surfaces :

$$\mathbf{M}_h(\mathbf{v}) = {}^{T}\mathbf{R} \cdot \begin{bmatrix} \frac{1}{\|\mathbf{v}\|^2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{\alpha^2} & 0 & 0 \\ 0 & \frac{1}{\beta^2} & 0 \\ 0 & 0 & \frac{1}{\beta^2} \end{bmatrix} \cdot \mathbf{R} \tag{130}$$

where **R** is a rotation matrix very similar to $\mathbf{R}_4$ introduced in equation (121) :

$$\mathbf{R} = \begin{bmatrix} \boldsymbol{\delta}_1^2 + (1-\boldsymbol{\delta}_1^2)\cos\theta & \boldsymbol{\delta}_1\boldsymbol{\delta}_2(1-\cos\theta) - \boldsymbol{\delta}_3\sin\theta & \boldsymbol{\delta}_1\boldsymbol{\delta}_3(1-\cos\theta) + \boldsymbol{\delta}_2\sin\theta \\ \boldsymbol{\delta}_1\boldsymbol{\delta}_2(1-\cos\theta) + \boldsymbol{\delta}_3\sin\theta & \boldsymbol{\delta}_2^2 + (1-\boldsymbol{\delta}_2^2)\cos\theta & \boldsymbol{\delta}_2\boldsymbol{\delta}_3(1-\cos\theta) - \boldsymbol{\delta}_1\sin\theta \\ \boldsymbol{\delta}_1\boldsymbol{\delta}_3(1-\cos\theta) - \boldsymbol{\delta}_2\sin\theta & \boldsymbol{\delta}_2\boldsymbol{\delta}_3(1-\cos\theta) + \boldsymbol{\delta}_1\sin\theta & \boldsymbol{\delta}_3^2 + (1-\boldsymbol{\delta}_3^2)\cos\theta \end{bmatrix} \tag{131}$$

In the above matrix, the rotation axis $\boldsymbol{\delta}$ is determined as follows:

$$\boldsymbol{\delta} = \frac{\overrightarrow{Ox} \wedge \mathbf{v}}{\left\| \overrightarrow{Ox} \wedge \mathbf{v} \right\|} \tag{132}$$

and θ is the angle of rotation determined by its sine and cosine:

$$\cos\theta = \frac{\overrightarrow{Ox} \cdot \mathbf{v}}{\|\mathbf{v}\|} \tag{133}$$

$$\sin\theta = \frac{\overrightarrow{Ox} \wedge \mathbf{v}}{\|\mathbf{v}\|} \cdot \boldsymbol{\delta} \tag{134}$$

The constants α and β in equation (130) are respectively a scale factor related to the norm of the velocity (it represents a time step between two layers of elements) and the density of elements in other directions than the velocity vector. They can be manually set or determined by an adequate global error estimator (for example, how many layers of elements between an injection runner and a vent). In particular, α can be set easily as it has a very explicit meaning. The factor β can be set to fit many purposes. One can set it to get an isotropic mesh, or to meet specified anisotropy factor. Also, it can be set directly by an error estimator.

A combined metric $\mathbf{M}_c(p, \mathbf{v})$ suited for the whole problem (diffusive and convective) is constructed to take the highest mesh density in each direction based on the metrics $\mathbf{M}_h(\mathbf{v})$ and $\mathbf{M}_c(p)$. Using the eigenvalues of $\mathbf{M}_h(\mathbf{v})$, it is easy to reconstruct a new metric with these requirements. Obviously, the matrix composed of the eigenvalues of $\mathbf{M}_h(\mathbf{v})$ is taken from its expression in equation (130), which are written for the three dimensional case :

$$\Lambda = \begin{bmatrix} \frac{1}{\|\mathbf{v}\|^2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{\alpha^2} & 0 & 0 \\ 0 & \frac{1}{\beta^2} & 0 \\ 0 & 0 & \frac{1}{\beta^2} \end{bmatrix} = \begin{bmatrix} \frac{1}{\|\mathbf{v}\|^2 \cdot \alpha^2} & 0 & 0 \\ 0 & \frac{1}{\beta^2} & 0 \\ 0 & 0 & \frac{1}{\beta^2} \end{bmatrix} \tag{135}$$

The expression of the compound metric becomes :

$$\mathbf{M}_c(p,\mathbf{v}) = {}^{\mathrm{T}}\mathbf{R}\cdot\begin{bmatrix} \max\left(\dfrac{1}{\|\mathbf{v}\|^2\cdot\alpha^2}, \dfrac{1}{\varepsilon}\max_i(|\lambda_i|)\right) & 0 & 0 \\ 0 & \max\left(\dfrac{1}{\beta^2}, \dfrac{1}{\varepsilon}\max_i(|\lambda_i|)\right) & 0 \\ 0 & 0 & \max\left(\dfrac{1}{\beta^2}, \dfrac{1}{\varepsilon}\max_i(|\lambda_i|)\right) \end{bmatrix}\cdot\mathbf{R} \qquad (136)$$

Like for the two-dimensional case, a global density factor η_i is defined for each loop of the algorithm to prevent the generation of over-refined meshes at early stages of the re-meshing loop, and prevent spurious oscillations while converging to the adapted mesh. The metric is modified as follows :

$$\mathbf{M}_c^{\eta_i}(p,\mathbf{v}) = \frac{\mathbf{M}_c(p,\mathbf{v})}{\eta_i} \qquad (137)$$

The initial value η_0 is given (we have chosen 5 in the latter example), but this parameter may be user-defined. Usually, for curved shells, the initial mesh must contain enough elements to represent correctly the geometry, so that one can choose a lower value but a good value has been found to be between 5 and 15. The following updating scheme is implemented as for two-dimensional case:

$$\begin{cases} \eta_{i+1} = \dfrac{\eta_i}{2} & \text{if } \eta_i > 2 \\ \eta_{i+1} = 1 & \text{if } \eta_i \le 2 \end{cases} \qquad (138)$$

6.4.6 Results of the mesh adaptation and discussion

The above formulas were implemented to control the mesh adaptation algorithm on a curved geometry. Figure 6.7-up shows the mesh of an ambulance roof manufactured by resin infusion under a flexible membrane. This is the base geometry used for the subsequent mesh adaptation (no reference is made to the original CAD model). The mould is considered completely filled and a Darcy problem is solved to obtain the pressure field and velocity of the resin in the mould under the normal manufacturing conditions. Then, these are used to generate an adapted mesh Figure 6.7-bottom) with the algorithms describes above. The new mesh is used to perform the real filling simulation shown in Figure 6.8-bottom. As a comparison, an isotropic mesh with approximately the same accuracy for this example (Figure 6.9) is shown.

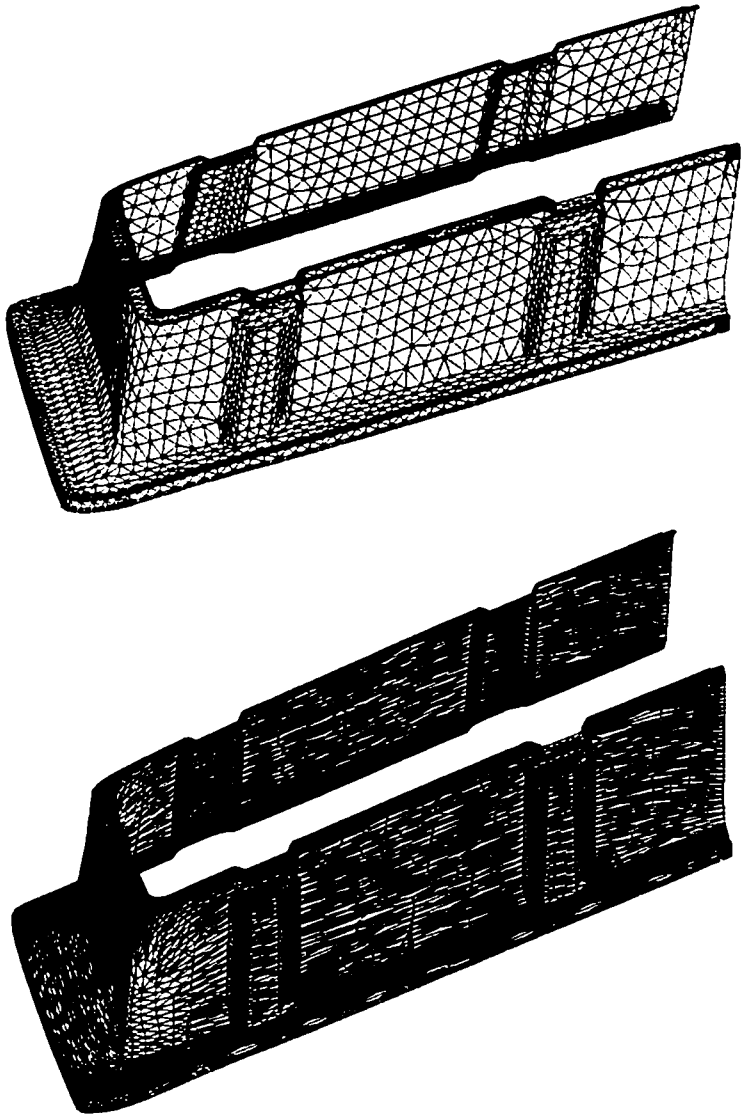


Figure 6.7: The original geometry (isotropic mesh generated with I-deas), and the mesh resulting of several stages of mesh adaptation carried out for a steady flow simulation. The injection runners are located on the top edge of the part, and the vent along the bottom edge. The original mesh has 7000 elements, and the adapted mesh, 11700 elements

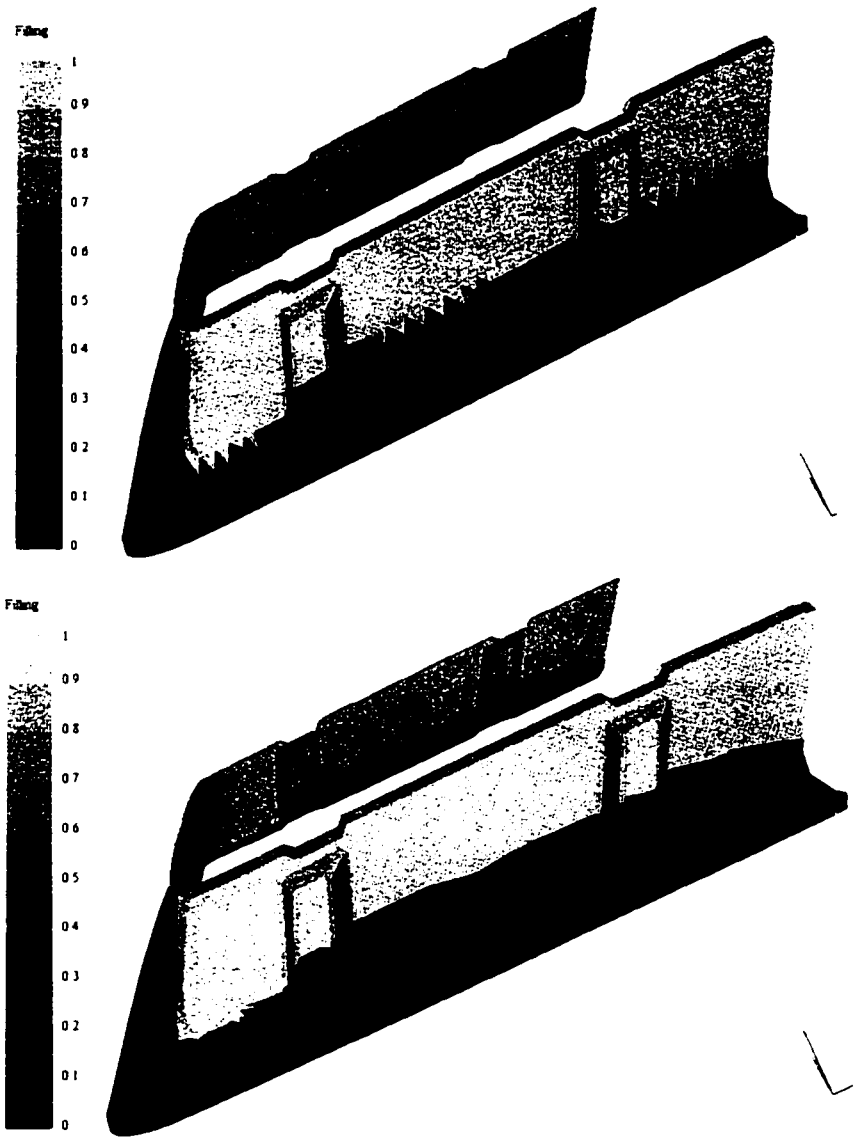


Figure 6.8: Simulations performed with the original mesh (top) and on the adapted mesh of Figure 6.7 (bottom). These figures show the filled part 30 seconds after the beginning of injection

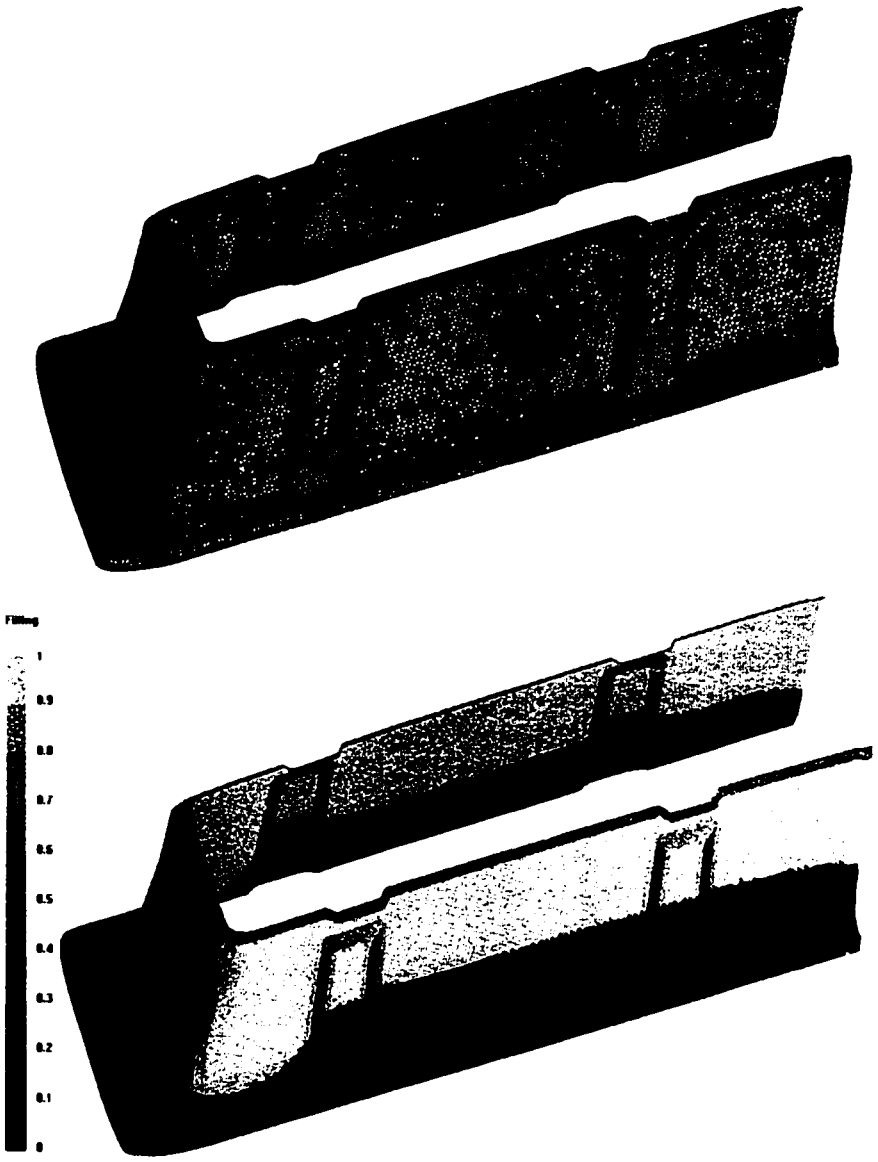


Figure 6.9: Simulation performed with a highly refined mesh, with the same accuracy as in the simulation with the adapted mesh. However, this mesh has 30700 elements

In this simulation, the results obtained with an adapted mesh containing around 11000 elements are comparable with those obtained with an isotropic mesh containing more than 30000 elements. When compared with results obtained with the initial mesh, it

shows narrower front, thus making thermal simulations more stable and accurate. In Part I, the same results of narrowing the diffusive region near the flow front were obtained. However, in that case, remeshing was made only around the flow front (each time step having a different mesh).

6.5 Conclusion

A remeshing procedure adapted to realistic RTM simulations is presented. First, the need for a specific error estimator is shown. Then, the expression of the hessian error estimator presented in Part I is modified to fit discrete surfaces encountered in finite element simulations. These are often used in RTM simulations because many industrial RTM parts are shells. Finally, a sample geometry from a real industrial part is used to demonstrate the gains that can be achieved with such an approach. Advantages are a lower computational requirement, and an improved front accuracy (at constant number of elements). Compared to the remeshing algorithm presented in [64], the drawback is that the mesh must be handled separately for adaptation purposes before the simulation takes place.

6.6 Acknowledgements

This work was supported financially by the National Science and Engineering Research Council of Canada (NSERC) and the Fonds Québécois de Recherche sur la Nature et les Technologies (FQRNT). A partial financial support was also received from Auto 21 to perform the validation for the ambulance roof. The authors also thank ESI-Group for the support concerning the software simulations performed with LCMFlot, and the companies Fibres Design (Chambly, Quebec) and Kaizen Technologies (Boucherville, Quebec) for allowing to publish the results of the numerical simulations performed for the ambulance roof. All these contributions are gratefully acknowledged.

6.7 References

[62] M. Ainsworth, J.T. Oden, 2000, A posteriori error estimation in finite element analysis. Wiley.

[63] E. Béchet. J.C. Cuillière. F. Trochu, 2002, Generation of a finite element mesh from stereolithography (STL) files, Computer Aided Design 34 , 1-17.

[64] E. Béchet. E. Ruiz. F. Trochu, J.C. Cuillière, 2003, Re-Meshing Algorithms Applied to Mould Filling Simulations in Resin Transfer Moulding, to appear in Journal of Reinforced Plastics and Composites.

[65] M.V. Bruschke. S.G. Advani. 1990. A Finite Element/Control Volume Approach to Mold Filling in Anisotropic Porous Media, Polymer Composites, **11** (6). 398-405.

[66] M.V. Brushke. S.G. Advani, 1991, Filling simulations of complex three dimensional shell-like structure, SAMPE Quarterly 2-11.

[67] G.C. Buscaglia. E.A. Dari, 1997, Anisotropic mesh optimization and its application in adaptivity. Int. J. Num. Meth. Eng. 40 (22), 4119-4136.

[68] P.P. Pébay. P.J. Frey. 1998, A priori Delaunay-conformity, Proceedings of the 7th International Meshing Roundtable, Dearborn (MI), 321-333.

[69] L. Piegl. 1991. On NURBS: A Survey, IEEE Computer Graphics and Applications 11 (1), 55 – 71.

[70] F. Trochu. 1993. A contouring Program Based on Dual Kriging Interpolation. Engineering with Computers 9,160-177.

[71] F. Trochu. R. Gauvin. D.M. Gao. 1993, Numerical Analysis of the Resin Transfer Molding Process by the Finite Element Method, Advances in Polymer Technology **12** (4) , 329-342.

[72] F. Trochu, P. Ferland, R. Gauvin, 1997, Functional Requirements of a Simulation Software for Liquid Molding Processes, Science & Engineering of Composite Materials 6 (4), 209-218.

[73] R. Verfürth, 1996, A review of a posteriori error estimation and adaptive mesh-refinement techniques, Wiley-Teubner (Chichester, Stuttgart).

[74] W.B. Young, K. Han, L.H. Fong, L.J. Lee, 1991,Flow Simulations in Moulds with Preplaced Fibre Mats, Polymer Composites **12** (6), 391-403.

[75] O.C. Zienkiewicz, J.Z. Zhu, 1987, A simple error estimator and adaptive procedure for practical engineering analysis, Int. J. Numer. Methods Eng. **24**, 335-357.

[76] O.C. Zienkiewicz, J.Z. Zhu, 1992, The superconvergent patch recovery and adaptive finite element refinement, Comput. Meth. Appl. Mech. Eng. **101** (1-3), 207-224.

CHAPITRE 7
DISCUSSION GÉNÉRALE

L'objectif général de cette thèse était de proposer une méthode de remaillage pour les problèmes à surface libre, au travers d'une application aux problèmes de remplissage de moules du procédé RTM pour les pièces en matériaux composites. En premier lieu, cet objectif a nécessité le développement d'un algorithme de maillage capable de travailler sur des surfaces courbes, sans avoir nécessairement l'information sur la surface exacte provenant de la CAO. L'idée était de pouvoir intégrer ce noyau de maillage dans un algorithme de remaillage par la suite. L'algorithme de maillage présenté ici est basé sur la méthode de Delaunay. Ce choix est dicté par le fait que l'on dispose à l'origine d'une triangulation de la surface. Dans le cadre restreint de l'article, il s'agit d'un fichier STL , et dans le cadre plus général de cette recherche, il s'agit d'un maillage initial fourni au début d'une simulation. Les autres méthodes ne sont pas applicables pratiquement car elles ne permettent pas de modifier continûment la triangulation ; il faudrait en effet reconstruire une surface analytique afin de les appliquer, ce qui est faisable mais peu efficace. Un autre critère est la rapidité puisque l'algorithme de remaillage doit être utilisé tout au long de la simulation; et dans ce cas, la méthode utilisée ici est un bon compromis avec une souplesse que les autres méthodes de maillage n'offrent pas. Les résultats de cette partie de la thèse (voir Chapitre 3) ont validé l'approche. Le point délicat est d'éviter de perdre de l'information sur la géométrie lors de la génération du maillage. La méthode de génération employée a été étendue au cas anisotrope en modifiant la façon avec laquelle les distances sont calculées. L'originalité de l'approche présentée ici est le postulat que l'on peut générer un maillage pour quelque application que ce soit à partir d'un fichier STL, que n'importe quel logiciel de CAO peut générer. Une question reste toutefois en suspens. En effet, l'évolution des logiciels de CAO tend à permettre une intégration toujours plus poussée entre les moyens de modélisation et les moyens de calcul. Pour ce faire, une

communication entre ces deux pôles est indispensable, par exemple en ce qui concerne les types de matériaux, les conditions aux limites, etc. Or, un fichier STL ne permet pas d'avoir d'information autre que géométrique. En l'état, une nouvelle saisie est donc nécessaire pour récupérer l'information perdue lors du transfert. Heureusement, ce n'est pas grave ; il est toujours possible d'adjoindre aux fichiers contenant la géométrie triangulée un fichier contenant de l'information supplémentaire qui serait alors utilisée directement dans la phase de calcul, et dans la phase de maillage (dans le cas de pré-optimisation de maillage).

Comme annoncé précédemment, les travaux de génération de maillage du Chapitre 3 sont utilisés afin de coder un algorithme de remaillage appliqué aux problèmes a frontière libre, en particulier pour le problème de remplissage en RTM. Pour ce faire, une carte de taille anisotrope a été construite pour chaque pas de temps, en fonction des conditions physiques locales. Cette carte de taille contient l'information sur la taille, l'allongement et l'orientation des éléments devant être générés. Le but étant de réduire la charge de calcul à précision constante, il fallait à tout prix éviter d'augmenter trop le nombre de pas de temps. L'évolution du front est donc gérée par une méthode de level-sets. Cette dernière permet de découpler l'algorithme d'avance de front et le maillage (en permettant de choisir le pas de temps). En effet, dans le code tel qu'implémenté, le pas de temps était choisi de façon à ne « remplir » qu'un nouvel élément à chaque pas de temps, ce qui induit une discrétisation temporelle très fine (et des simulations plus longue). L'originalité de l'approche présentée ici est le couplage entre une méthode de remaillage (permettant d'augmenter la précision des simulations dans le voisinage du front) et une méthode d'évolution de front par level-sets (permettant un contrôle « utilisateur » du pas de temps), ainsi que l'application de ces approches à la simulations du procédé RTM. Dans le Chapitre 4, les résultats sont présentés et montrent que l'on peut réduire le nombre de pas de temps d'un facteur 4, sans entacher notablement d'erreur le résultat du calcul. Mais savoir si l'on gagne également un facteur 4 dans le temps de calcul n'est pas évident à formuler car le remaillage a un coût non négligeable.

Toutefois, les gains constatés dans la régularité du front et du profil de pression dans son voisinage restent sans commune mesure avec ce que l'on pourrait obtenir en faisant un raffinement de maillage isotrope et uniforme. Dans ce dernier cas, la durée du calcul serait beaucoup plus longue. Le nombre d' éléments croît en effet avec le carré de la précision demandée (pour une simulation bidimensionelle), alors qu'il ne devrait croître en théorie que proportionnellement dans le cas où des éléments anisotropes sont utilisés.

Une extension au cas de simulations avec phénomènes de transport est présentée par la suite. Un aspect considéré ici est le comportement de la méthode numérique assurant le transport dans le cadre de phénomènes thermiques, en RTM, par exemple. Dans ce dernier cas, la température et le taux de conversion de la résine (lors de la polymérisation) doivent être transportés; et il est alors prouvé que l'on ne peut choisir arbitrairement le pas de temps. Une condition sur le nombre de Courant est imposée : celui ci ne doit en principe pas dépasser l'unité. En termes physiques, cela signifie que l'on ne peut pas transporter la grandeur scalaire de plus d'une couche d'éléments a la fois. Dans le cas contraire, la méthode numérique utilisée ici (Lesaint-Raviart pour le transport suivie de Galerkin standard pour la diffusion) n'est pas stable et l'on voit apparaître des oscillations (voir Chapitre 5). Dans le cas du remaillage tel qu'appliqué dans le Chapitre 4, le pas de temps est généralement trop grand pour assurer la stabilité du schéma de transport. L'idée retenue ici pour éviter des résultats erronés est d'effectuer plusieurs calculs de transport entre deux « grands » pas de temps du remaillage. Ceci évite de remailler à chaque pas de temps thermique, et assure la stabilité du schéma. Toutefois, le fait d'avoir à calculer plusieurs pas de temps thermiques à l'intérieur d'un pas de temps de remplissage oblitère les gains en temps de calcul que l'on peut attendre du remaillage, d'autant plus que les calculs de transport et de diffusion thermiques sont grands consommateurs de ressources. En revanche, un net gain est visible pour ce qui est de la diffusion numérique (diffusion parasite liée au schéma numérique). Cette dernière est en effet proportionnelle à la taille des éléments

utilisés pour résoudre l'équation de transport, et ceux ci sont aplatis dans le sens de l'écoulement au voisinage du front.

Afin de minimiser l'impact du remaillage en cours de calcul, il est toujours possible de tenter de générer un maillage adapté *a priori* à la simulation complète. L'idée est de pouvoir tenir compte de deux erreurs : l'erreur d'interpolation de la solution en pression du problème de Darcy, et l'erreur liée à l'algorithme de remplissage pour la mise à jour du front à chaque pas de temps. Ceci implique d'une part de générer des éléments allongés en fonction de la vitesse du front « au moment ou il passera a cet endroit » et d'autre part de tenir compte d'un estimateur d'erreur classique pour l'erreur d'interpolation. Dans le cas des surfaces courbes, il est montré que l'estimateur n'est pas applicable tel quel car il inclut l'erreur de discrétisation géométrique. Cette dernière est en effet une donnée du problème (dépendante de ce qui est fait en relation avec le Chapitre 3). L'adaptation faite à l'estimateur d'erreur consiste à considérer la géométrie localement plane et a effectuer une rotation correspondante des vecteurs gradient afin de les faire coïncider avec le plan tangent. Il est à noter que ceci permet aussi de faire un calcul d'erreur d'interpolation au voisinage d'une arrête « réelle » d'un solide, sans être faussé par le fait que les éléments utilisés pour le calcul d'erreur ne sont pas coplanaires. Les calculs montrés dans le Chapitre 5 et le Chapitre 6 tendent à montrer la validité de cette approche. Toutefois, l'étude d'un cas analytique d'injection centrale en montre les limites. Pour ce cas, il est possible de générer un maillage adapté *a priori* en considérant deux contraintes distinctes : soit on force l'erreur d'interpolation à être constante dans le domaine, soit on se place dans le cas où l'on cherche a rendre le nombre de Courant constant dans le domaine. La comparaison des deux maillages obtenus est éloquente : on ne peut pas à la fois satisfaire les deux conditions, car une faible erreur d'interpolation impose de diminuer la taille du maillage, en opposition avec le fait que pour diminuer le nombre de Courant, il faut augmenter la taille des éléments dans le sens de l'écoulement, ou diminuer le pas de temps (ce que l'on cherche à éviter). Pour ce cas analytique, il n'existe pas de maillage fixe optimal en termes de calcul. Le remaillage

semble donc nécessaire, au moins théoriquement. En pratique, il est toujours possible de nuancer cette affirmation si l'on évite de prendre des pas de temps trop grands, au prix de temps de calcul plus longs.

CHAPITRE 8
RECOMMANDATIONS

Quelques recommandations sont énoncées dans cette partie. Elles visent à proposer quelques idées pour poursuivre la recherche initiée dans cette thèse.

- Les résultats présentés ici ont été menés pour des géométries planaires et surfaciques (courbes). Une extension de ce travail dans le cas tridimensionnel est souhaitable : il n'y a pas d'obstacles théoriques à cela si ce n'est la plus grande complexité des algorithmes de maillage tridimensionnels. Dans la lignée du Chapitre 3, la construction d'un maillage tridimensionnel respectant une carte de taille à partir d'un fichier STL est proposée.

- L'approche par level-sets proposée ici pour l'évolution du front de résine dans le cadre du RTM est très souple; elle doit être étendue dans le cas tridimensionnel. Toutefois, pour des raisons d'ordre théorique, il serait avantageux de considérer une autre formulation que celle d'Hamilton-Jacobi. Cette formulation alternative est présentée dans (Gomes, 1999). Cette dernière permettrait de rendre compte de phénomènes plus complexes qu'un écoulement en milieu poreux (écoulements ne dérivant pas d'un potentiels, par exemple avec effets dynamiques).

- L'imposition des conditions aux limites en pression sur la frontière libre est faite de façon approchée aux nœuds immédiatement voisins. Ceci induit une légère erreur au sens fonctionnel sur la position réelle de la frontière. Il serait avantageux d'utiliser une approche permettant d'imposer cette condition précisément même si la surface libre n'est pas exactement à la frontière entre

deux éléments. Ceci est possible en ajoutant des multiplicateurs de Lagrange pour chaque élément contenant la surface libre, ou par une technique d'enrichissement des éléments finis (Moës, 2000), entre autres.

- Dans le cas de moulage par RTM de pièces en matériaux composites comportant un insert interne (p.ex. intérieur en mousse d'une planche à voile), il arrive fréquemment qu'il y ait déplacement de ce dernier si la progression du front de résine n'est pas uniforme. Ce déplacement induit a son tour des modifications de la perméabilité des fibres l'entourant. Un champ de recherche encore ouvert est la simulation complète de ce déplacement. Si le déplacement est grand, il sera en effet nécessaire de remailler de façon à conserver un maillage conforme pour pouvoir mener la simulation à son terme. Les level-sets peuvent être avantageusement utilisés pour permettre ce remaillage et représenter le déplacement de l'insert sur un maillage qui reste globalement eulerien. Une autre approche pourrait de passer à une formulation lagrangienne pour le déplacement de l'insert..

- Dans le cas de simulations thermiques, tenir compte d'un estimateur d'erreur sur la température, en plus de celui sur la pression, permettrait certainement de contrôler mieux la précision. Toutefois, il faut aussi tenir compte du nombre de Courant. Une analyse plus poussée est souhaitable dans ce cas particulier.

- La simulation des phénomènes de transport est faite avec une approche purement Eulerienne. Deux conséquences en découlent : la présence de diffusion numérique, et la contrainte CFL sur le pas de temps qui doit être respectée. En remplaçant l'approche Eulerienne par une approche semi-Lagrangienne, il est possible d'effectuer des simulations moins diffusives et à plus grand pas de temps. Ceci permet d'exploiter les avantages liés à l'utilisation d'algorithmes de remaillage et level-sets sans perdre au niveau du temps calcul. Dans le cas de la

simulation de phénomènes thermiques et de cuisson en RTM, cela permet de rendre les simulation thermiques moins dépendante du maillage utilisé pour résoudre l'écoulement de Darcy. Il faut toutefois vérifier que l'utilisation de l'approche semi-Lagrangienne proposée ici ne se fait pas au détriment de la conservation de la quantité transportée; ce qui constitue un inconvénient documenté de la méthode semi-Lagrangienne. Il serait intéressant d'étudier la valeur maximale du pas de temps autorisé afin que la conservation soit vérifiée dans des proportions admissible pour le cas de figure traité ici.

CONCLUSION

Cette thèse a montré l'utilité des méthodes de remaillage appliquées à l'évolution de surfaces libres. Une étude préliminaire sur le maillage de surfaces discrète a été présentée. Celle ci montre la faisabilité d'un mailleur découplé de la géométrie exacte telle que définie dans la CAO. Une extension naturelle aux maillages anisotropes a été faite. L'application aux simulations de remplissage de moules par RTM (Resin Transfer Moulding) nécessite en effet de considérer des surfaces discrètes de façon à éviter l'interaction avec la CAO qui est à l'origine du modèle géométrique exact (inconnu en pratique). Les problèmes de surfaces libres sont habituellement traités sur un maillage fixe et invariant dans le temps (Eulerien), ce qui peut poser des problèmes d'approximation au voisinage de la frontière mobile. Une approche originale de remaillage adaptatif associé à la formulation de l'évolution de la surface libre par level-sets est présentée ici, et une validation tant expérimentale que numérique est effectuée. Les simulations d'écoulement impliquent souvent le transport de grandeurs scalaires (température, etc.) dans le domaine. Une application du remaillage adaptatif est effectuée pour un cas de simulation thermique et une condition sur le pas de temps (condition CFL) est nécessaire afin d'assurer la stabilité du schéma de transport. Par la suite une adaptation de maillage effectuée *a priori* est proposée. Ceci permet de n'utiliser qu'un seul maillage pour toute la simulation. L'adaptation est faite en tenant compte de deux critères : erreur d'interpolation et nombre de Courant. Une étude théorique démontre ensuite la difficulté de combiner les deux dans certains cas (injection centrale). L'estimateur d'erreur présenté ici est basé sur le Hessien d'une variable scalaire pour la partie « erreur d'interpolation ». Une extension est proposée dans le cas des surfaces courbes, visant à découpler l'erreur géométrique de l'erreur fonctionnelle. Cette dernière est utilisée seule, car l'erreur géométrique est une donnée du problème et ne peut à ce stade être diminuée. La validation de cette approche est ensuite faite pour

une pièce industrielle complexe qui montre que l'on peut énormément gagner en nombre d'éléments par rapport à une simulation à maillage isotrope.

BIBLIOGRAPHIE

BÉCHET, É, CUILLIÈRE, J.C. , REMACLE, J.F. et TROCHU, F. (1999). Transformation d'un maillage "STL" pour les méthodes numériques, Actes de la Journée Maillage du CERCA.

BONET, J. et PERAIRE, J. (1991). An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems, Intl. J. for Num. Meth. in Eng, 31, 1-17.

BOROUCHAKI, H., GEORGE, P.L., HECHT, F., LAUG, P. et SALTEL, E. (1997). Delaunay mesh generation governed by metric specifications. Part I : Algorithms, F.E. in Analysis and Design, 25, 61-83.

BOROUCHAKI, H., GEORGE, P.L., HECHT, F., LAUG, P. et SALTEL, E. (1997). Delaunay mesh generation governed by metric specifications. Part II : Applications, F.E. in Analysis and Design, 25, 85-109.

CAMARERO, R. et REGGIO, M. (1983), A multigrid scheme for 3-D body fitted coordinates in turbomachinery applications, ASME J. of fluids engineering, 105, 76.

CAUCHOIS, J.P. (1997), R.T.M. Process, Éditions Syntech.

CAVENDISH, J.C. (1974), Automatic triangulation of arbitrary planar domains for finite element methos, Intl. J. For Num. Meth. In Eng, 8, 679-696.

CAVENDISH, J. C., FIELS, D.A et FREY, W.H. (1985), An approach to automatic three-dimensional finite element mesh generation, Intl. J. for Num. Meth. In Eng, 21, 329-374.

CHAE, S.W. et BATHE, K.J. (1989), On automatic mesh construction and mesh refinement in finite element analysis, Computer & Structures, 32, 911-936.

CIGNONI, P., MONTANI, C. et SCOPIGNO, R. (1998), DeWall: a fast divide & conquer Delaunay triangulation algorithm in Ed. Computer-Aided Design, 30, (5), 333-341.

CHERFILS, C. et HERMELINE, F. (1990), Diagonal swap procedures and characterizations of 2D-Delaunay triangulations, Math. Mod. And Num. Anal., 24, (5), 613-626.

COLLINO, F., FOUQUET, T. et JOLY, P. (1998), Analyse numérique d'une méthode de raffinement de maillage espace-temps pour l'équation des ondes, R.R. INRIA No. 3474.

COUPEZ, T. (1991), Grandes transformations et remaillage automatique, Thèse de l'ENSMP, Paris.

CUILLIÈRE, J.C. (1997), A direct method for the automatic discretization of 3D parametric curves, Comp. Aided Design, 29, (9), 639-647.

CUILLIÈRE, J.C. (1998), An adaptive method for the automatic triangulation of 3D parametric surfaces, Comp. Aided Design, 30, (2), 139-149.

DELAUNAY, B.N. (1934), Sur la sphère vide, Bul. Acad. Sci. URSS, Class. Sci.Nat., 793-800.

DIRICHLET, G.L. (1850), Über die Reduktion der positiven quadratischen Formen mit drei understimmten ganzen zahlen, Z. Angew. Math. Mech., 40,(3), 209-227.

FRANÇOIS, V. (1998), Méthodes de maillage et de remaillage automatiques appliquées à la modification de modèle dans le contexte de l'ingénierie simultanée, Thèse de l'université Henri Poincaré, Nancy I.

FREY, P.J. et MARECHAL, L. (1998), Fast adaptive quadtree mesh generation, proceedings of the 7th International Meshing Roundtable, 211-224.

GEORGE, P.L. et BOROUCHAKI, H. (1997), Triangulation de Delaunay et maillages : application aux éléments finis, Paris, Hermès. ISBN 2-86601-625-4.

Golgolab, A. (1989), Mailleur 3D automatique pour des géométries complexes, R.R. INRIA No. 1004.

GOMES, J. et FAUGERAS, O. (1999), Reconciling distance function and level-sets, R.R. INRIA No. 3666.

GUIBAULT, F. (1998), Un mailleur hybride structuré/non structuré en trois dimensions, Thèse de doctorat, École polytechnique de Montréal.

HASSAN, O., PROBERT, E.J. et MORGAN, K. (1998), Unstructured mesh procedures for the simulation of three-dimensional transient compressible inviscid flows with moving boundary components, Int. J. Numer. Meth. Fluids, 27, 41-55.

LEBRUN, G. (1995), Étude des phénomènes d'échanges thermiques pour le moulage par transfert de résine, Thèse de doctorat, École Polytechnique de Montréal.

LE ROUX, D.Y., LIN, C.A. et STANIFORTH, A. (1997), An Accurate Interpolating Scheme for Semi-Lagrangian Advection on an Unstructured Mesh, Tellus series A, 49, (2),119-138.

LI, X.Y., TENG, S.H. et ÜNGÖR, A. (1998), Simultaneous refinement and coarsening: Adaptive Meshing with moving Boundaries, Proceedings of the 7th International Meshing Roundtable, 201-210.

LO, S.H. (1991), Volume discretization into tetrahedra-I. Verification and orientation of boundary surfaces, Computers and Structures, 39, (5), 493-500.

LO, S.H. (1991), Volume discretization into tetrahedra-II. 3D triangulation by advancing front approach, Computers and Structures, 39, (5), 501-511.

LÖHNER, R. et PARIKH, P. (1988), Generation of three-dimensional unstructured grids by the advancing-front method, Int. J. num. meth. Fluids, 8, 1135-1149

MANTYLA, M. (1988), An introduction to solid modeling, Computer science press.

MILLER, G.L., TALMOR, D. et TENG, S.H. (1999), Optimal coarsening of unstructured meshes, Journal of Algorithms, 31, 29-65.

MOËS, N. (2000), Contribution au calcul des structures : une extension de la méthode des éléments finis ; le contrôle des calculs en éléments finis non linéaires, mémoire d'habilitation à diriger des recherches, Université Paris 6.

MÖLLER, P. et HANSBO, P. (1995), On advancing front mesh generation in three dimensions, Intl. J. num. meth. eng., 38, 3551-3569.

OSHER, S. et SETHIAN, J.A. (1988), Front propagating in with curvature dependent speed : algorithms based on Hamilton-Jacobi formulations, Journal of Computational Physics, 79, (1), 12-49.

PÉBAY, P.P. et FREY, P.J. (1998), A priori Delaunay-conformity, Proceedings of the 7th International Meshing Roundtable, 321-333.

PÉRAIRE, J., PEIRO, J., FORMAGGIA, L. et ZIENKIEWICZ, O.C. (1988), Finite Element Euler computation in three dimensions, Int. J. numer. meth. eng. ,26, 2135-2159.

RASSINEUX, A. (1995), Maillage automatique tridimensionnel par une méthode frontale pour la méthode des éléments finis, Thèse de l'université Henri Poincaré, Nancy I.

RASSINEUX, A. (1997), 3D mesh adaptation. Optimization of tetrahedral meshes by advancing front technique, Comp. Meth. Appl. Mech. Engrg., 141, 335-354.

REMACLE, J.F., BÉCHET, E. et TROCHU, F. (1999), Maillages contrôlés isotropes et anisotropes des surfaces paramétriques, Actes de la Journée Maillages du CERCA.

RIVARA, M.C. et INOSTROZA, P. (1997), Using longuest-side bisection techniques for the automatic refinement of Delaunay triangulations, Intl. J. for Num. Meth. in Eng., 40, 581-597.

RUDD, C.D., LONG, A.C., KENDALL, K.N. et MANGIN, C.G.E. (1997). Liquid Moulding Technologies, Woodhead Pub. Ltd.

SCHÖNHART, E. (1928), Über die Zerlegung von Dreieckspolyedern, Mathematische Annalen, 98, 309-312.

SCHUTZ, B.F. (1980). Geometrical methods of mathematical physics, Cambridge University Press.

SHIMADA, K. (1993). Physically-based mesh generation: Automated triangulation of surfaces and volumes via bubble packing, Ph.D. Thesis, Massachusetts Institute of Technology.

SHIMADA, K. et GOSSARD, D. (1995), Bubble mesh: automated triangular meshing of non-manifold geometry by sphere packing, ACM Third Symposium on Solid Modeling and Applications, 409-419.

SHIMADA, K. YAMADA, A. et ITOH, T. (1997), Anisotropic triangular meshing of parametric surfaces via close packing of ellipsoidal bubbles, Proceedings of the 6th International Meshing Roundtable.

THACKER, W.C., GONZALES, A. et PUTLAND, E. (1980), A method for automating the construction of irregular computational grid for storm surge forecast models, J. Computational Physics, 37, 371-387.

THOMSON, J.F., WARZI, Z.U.A et MASTIN, C.W. (1985), Numerical grid generation, North-Holland.

TRISTANO, J.R., OWEN, S.J. et CANANN, S.A. (1998), Advancing front mesh generation in parametric space using a Riemannian surface definition, Proceedings of the 7th International Meshing Roundtable, 429-445.

VALLET, M.G. (1992), Génération de maillages éléments finis anisotropes et adaptatifs, Thèse de doctorat de l'université paris VI.

VORONOÏ, G. (1908), Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Recherches sur les parallélоèdres primitifs. Journal Reine angew. math., 134.

WATSON, D.F. (1981), Computing the n-dimensional Delaunay Tesselation with application to Voronoï polytopes, Computer Journal, 24, (2), 167-172.

WINSLOW, A. (1967), Numerical solution of the quasilinear Poisson equation in a nonuniform mesh, Journal of Computational Physics, 2, 149-172.

YERRY, M.A. et SHEPHARD, M.S. (1984), Automatic Three-Dimensional Mesh Generation By The Modified Octree Techniques, International Journal For Numerical Methods in Engineering, 20, 1965-1990.

ZWART, P.J. et RAITHBY, G.D. (1998), Space-time meshing for two-dimensional moving boundary problems, proceedings of the 7th International Meshing Roundtable, 187-200.

ANNEXE 1
RÈGLES SÉMANTIQUES DE LA LIBRAIRIE DE MAILLAGE

1.1 Introduction

Il n'est pas inutile de mettre le doigt sur les conventions utilisées lors de l'implémentation de la librairie de maillage. En effet, celles ci sont inspirées d'un paradigme de programmation, appelé « programmation générique », qui fut à l'origine d'une des abstractions les plus fécondes en programmation C++. Cette façon de programmer n'est pas à proprement parler exclusive au C++, elle a toutefois été rapidement intégrée dans ce langage car elle répondait à un certain besoin. Une partie de la plus récente norme C++ à ce jour [77] est basée entièrement sur ce paradigme; il s'agit de la « standard template library » (STL par la suite) [79]. Cette librairie n'est en fait qu'une formalisation de la notion de structure de donnée (dans le sens premier d'objet qui sert à contenir des données). Cette formalisation est construite à l'aide de toutes les possibilités offertes par le langage C++ : la programmation orientée objet (POO) bien sur, mais aussi et surtout la notion de classe patron (template class) [78]. Ainsi, les concepteurs de la STL ont réussi à rendre leur librairie totalement abstraite, en ce sens que mis à part quelques contraintes sur les objets que cette librairie doit manipuler, elle ne dépend pas de leur implémentation précise. Les contraintes sur les objets sont liées précisément à la nature de la STL : stocker de diverses façons l'information. On peut ajouter qu'elles ne sont pas superflues, mais plutôt nécessaires. Par exemple, il faut bien définir une relation d'ordre entre objets si on veut les classer (selon cette même relation d'ordre). Si le but n'est pas de classer les objets, alors il n'est pas nécessaire de fournir de relation d'ordre. On peut conclure par cette phrase : « La finalité d'une construction de programmation générique impose quelques contraintes minimales sur les types d'objets manipulés ».

1.2 Exemple d'utilisation de la STL

La STL est extrêmement simple d'utilisation, c'est pourquoi il est de bon ton de la présenter avant de passer à la librairie de maillage, qui, comme indiqué dans l'introduction, est basée sur le même paradigme. Il est bon de mentionner deux entités importantes qui sont à la base de la philosophie utilisée dans la STL. Par la suite, quelques exemples d'utilisation sont exposés.

1.2.1 Container

Un « container » est un genre de contenant d'information dans la STL. Ces derniers sont de plusieurs types et comme on peut s'en douter, il existe plusieurs containers adaptés a chaque utilisation particulière, et pas un container adapté à toutes les sortes d'utilisation. Ils sont principalement au nombre de 7 : `vector`, `list`, `set`, `multiset`, `map`, `multimap`, `deque`, mais rien n'empêche le développement futur d'autres types dans de futures révisions de la norme C++. Ils prennent la forme suivante si l'on passe comme paramètre de template le type du contenu `T`:

```
type_du_container<T> nom_du_container ;
```

Pour une description exhaustive chacune de leurs caractéristiques, voir [79] ou [80].

1.2.2 Itérateur

Dépendamment de la nature du container, le parcours dans celui ci est restreint à certaines opérations uniquement. Pour une liste (`list`), on ne peut que passer d'un élément à un autre immédiatement suivant ou précédent (accès séquentiel). Pour un vecteur (`vector`), l'accès est aléatoire. Par conséquent, dans la STL, il est nécessaire

construire des pointeurs un peu particuliers permettant / restreignant le parcours selon le type de container. Ce sont les itérateurs (`iterator`). Ils s'utilisent comme les pointeurs en C/C++ , mais ,de plus, permettent les opérations de parcours associées à chaque container. Généralement, les opérateurs suivants sont définis sur les pointeurs, tout comme sur les itérateurs :

`*I`	: déréférencement (accés à la donnée pointée)
`++I` ou `I++`	: élément suivant
`--I` ou `I--`	: élément précédent
`I[k]`	: accès direct à l'élément suivant `k` fois l'élément pointé par `I`.

Il faut néanmoins noter que pour certains containers, seules certains opérateurs sur les itérateurs associés sont définis. En particulier, il n'existe pas d'accès direct pour les listes chaînées, et donc pas d'opérateur « `[]` ». Pour conclure, les itérateurs constituent de fait une généralisation de la notion de pointeur en C, permettant une syntaxe très simple lors de la manipulation des classes de la STL. Ils sont déclarés comme des types associés au container, de ce fait on peut récupérer leur type par une syntaxe du style :

```
container<T>::iterator
```

1.2.3 1er exemple : une liste chainee

Il existe dans la STL la notion de liste chaînée. Une liste chaînée est une structure dans laquelle les objets sont stockés en chaîne, les uns après les autres. Il n'est possible d'accéder à un objet de la liste qu'en la parcourant depuis le début (1er objet). On parle d'accès séquentiel. La liste se déclare dans le cas général :

```
list<T> ma_liste;
```

où T est le « paramètre de template », i.e. le type d'objets que la liste « ma_liste » est censée contenir. Comme tout objet, « ma_liste » est muni de fonctions membres. Les plus importantes ici sont reliées à la nature de liste chaînée : insertion d'éléments en 1ère position, en fin de liste, derrière ou devant un élément identifié de la liste (respectivement fonctions push_front(), push_back(), insert()). Délétion d'éléments dans les mêmes conditions (respectivement fonction pop_front(), pop_back(), delete()), etc. . Les contraintes sur T sont assez restreintes pour une liste chaînée ; ils doivent simplement être assignables (posséder l'opérateur « = »). N'importe quel objet assignable peut être membre, en particulier tous les types par défaut du C++ (intégraux int, long, char..., réels float, double, ...).

Le petit programme suivant insère une séquence croissante de nombre entiers dans la liste et l'affiche à l'envers :

```
list<int> ma_liste;           // une liste d'entiers
for (int i=0 ; i<15 ; ++i)    // pour les 15 premiers...
{
      ma_liste.push_front(i); // insertion au debut
      cout << I << " ";       // affichage
}
cout << endl;
list<int>::iterator it        // un iterateur
it=ma_liste.begin();          // au debut de la liste
while (it!=ma_liste.end())    // tant que pas a la fin
{
      cout << *it << " ";     // affichage de l'element
      ++it;                   // passage a l'element suivant
}
cout << endl;
```

affichage :

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
```

1.2.4 2nd exemple : une entité de classification

Il est possible dans le cadre de la STL d'utiliser un des container (`set`, `multiset`, `map`, `multimap`) qui conserve en permanence une classification selon un certain critère de leur contenu. Le `set` est le plus simple, c'est une séquence ordonnée unique (i.e. chaque élément est unique). En passant le type d'objet `T` en paramètre de template, il se déclare simplement :

```
set<T> ma_classif;
```

Le container `set` est muni de certaines opération d'insertion/délétion, dont `insert()`, `erase()`, mais contrairement à la liste chaînée, l'ordre d'insertion n'a pas de conséquence sur l'ordre dans lequel les éléments sont stockés. C'est la présence de l'opérateur « < » entre éléments de type `T` qui induit un ordre dans le `set`. Ainsi, le type paramètre T doit comporter un opérateur « < » afin de pouvoir être utilisé avec n'importe lequel des containers cités plus haut. Il s'agit de la contrainte évoquée dans l'introduction. En plus de celle ci, l'objet doit être assignable, donc posséder l'opérateur « = ». L'exemple suivant montre une possible utilisation, ou il s'agit de classer des entiers a la volée, en interdisant les doublons.

```
set<int> ma_classif ;          // un set d'entiers
for (int i=0 ;i<15 ;i++)
{
      int elem=rand()%10;      // nb au hasard entre 0 et 10
      cout << elem << " " ;    // on l'affiche
      ma_classif.insert(elem); // insertion dans le set
}
cout << endl ;
set<int>::interator it ;       // un iterateur ad-hoc
it=ma_classif.begin();         // au debut
while(it!=ma_classif.end())    // tant que pas a la fin
{
      cout << *it << " ";      // affiche l'entier
```

```
      ++it;                    // element suivant
}
cout << endl ;
```

affichage :

```
9 5 7 9 1 1 5 8 3 3 4 5 8 0 4
0 1 3 4 5 7 8 9
```

On voit bien que la seconde liste de nombre contient tous les éléments de la première, classés dans l'ordre défini par l'opérateur « < » sur les entiers (croissant), et ne contient chaque élément qu'une seule fois (d'ou sa taille réduite).

1.3 Structures de maillage

Les structures de maillages considérées ici se sont inspirées de la philosophie de la programmation générique dont la STL brièvement présentée plus haut est le meilleur représentant. Le but est de définir une classe template et tous les « accessoires» qui vont avec afin de représenter une géométrie de maillage.

Un maillage constitue une relation topologique entre des nœuds d'une part, et des éléments. De façon interne, il est possible de gérer d'autres entités pour diverses raisons (parcours rapide des relations de voisinage entre éléments, raffinement de maillage, etc.) Les entités de base sont les nœuds et les éléments, ce sont eux qui « génèrent » les caractéristiques d'un maillage. Nous allons donc les décrire succinctement. Elles seront construites au fur et a mesure, ainsi, le nœud ne peut être défini que si l'on connaît sa finalité. En conséquence, il faut lire séquentiellement les parties qui suivent.

1.3.1 Les nœuds

Un nœud est un point défini dans l'espace, typiquement $\Re^n$, n étant la dimension de l'espace. Pour décrire un nœud, il faut donc spécifier dans quel espace il se trouve, et chacune des coordonnées qui lui donnent sa position. La classe template suivante fait pour le moment l'affaire :

```
template<int dim> class vertex;
```

Le paramètre entier `dim` sert à donner une dimension à l'espace d'accueil du nœud. De façon interne, la classe `vertex` stocke les coordonnées dans un tableau de taille fixe (peu importe comment, à ce niveau). L'espace d'accueil n'est pas nécessairement euclidien : il peut s'agir d'un manifold quelconque (un patch en CAO par exemple), auquel cas les coordonnées sont données dans une base locale.

1.3.1.1 Contraintes sur les paramètres de template

Le paramètre `dim` doit être un entier strictement positif.

1.3.2 Les éléments

1.3.2.1 Élément générique

Par élément il est entendu un groupe de nœuds et des connectivités permettant d'identifier une partie finie de l'espace (« élément fini »). Donc, l'élément dépend de l'espace d'accueil et du type de nœud. La declaration est la suivante :

```
template<int dim, class V=vertex<dim> > class element;
```

le paramètre V possède un paramètre template par défaut « vertex<dim> ». Il est parfaitement possible de déterminer un autre type de nœud, en le faisant dériver de vertex et l'utiliser à sa place. L'élément ne contient pas les nœuds en tant que tels, mais il y réfère par l'intermédiaire d'itérateurs (ou de pointeurs). Ceci a une implication sur la définition des nœuds comme décrit ci-après.

1.3.2.1.1 Contraintes sur les paramètres de template

Le paramètre de template « dim » doit être strictement positif. V doit être une classe assignable, et doit contenir le membre Iterator. (pointeur/itérateur sur un objet du type de V). Cette contrainte s'ajoute donc à la définition du nœud au §1.3.1, et la modifie ainsi :

```
template<int n> class vertex
{
...
public :
      typedef vertex<n> *Iterator;  // pointeur (EXEMPLE)
...
};
```

1.3.2.2 Simplexes

Dans la plupart des cas, la génération de maillage s'intéresse à des éléments simplexes pour d'évidentes raisons de simplicité et de généralité. Un *d-simplexe* est un élément comportant d+1 nœuds et évoluant dans un espace de dimension au moins égale à d. Il s'agit de l'élément le plus simple constructible avec d+1 nœuds. Exemples : le triangle dans le plan (2-simplexe dans un espace de dimension 2), le tétraèdre (3-simplexe dans l'espace de dimension 3), un segment de droite dans le plan (1-simplexe), etc.. Comme le type élément, le type simplexe est construit à l'aide du nœud et de la dimension de

l'espace d'accueil. Il comporte en plus le nombre de nœud qui le définit, puisque celui ci est indépendant de la dimension de l'espace, mais en aucun cas supérieur à d+1. La définition est la suivante :

```
template<int n,int d,class V=Vertex<d> > class Simplex
                                                : public element<d,V> ;
```

Le paramètre de template `n` est le nombre de nœuds du simplexe, `d` est la dimension de l'espace, et `V` est le type de nœud considéré (même remarques que dans le §1.3.2.1).

1.3.2.2.1 Contraintes sur les paramètres de template

Le paramètre `d` est un entier strictement positif, `n` est supérieur à 2 et inférieur à `d+1`. La classe `V` doit respecter les mêmes conditions qu'au §1.3.2.1.1.

1.3.3 Faces topologiques

Cette entité est destinée à formaliser les relations de topologie entre les éléments d'un maillage (relation élément - élément). Il s'agit d'une face commune a deux éléments (ou plus dans le cas de géométries non-manifold). La face est identifiée par ses nœuds, et accessoirement contient l'information sur le ou les éléments qui s'y rattachent. La déclaration est la suivante :

```
template<class Vit,class Eit> class Topological_Face
                                    : public Topology_Entity<Vit,Eit> ;
```

Cette classe prend deux paramètres de template. Le type `Vit` est de type itérateur sur les nœuds, et `Eit` l'est sur des éléments.

1.3.3.1 Contraintes sur les paramètres de template

Ils doivent être des itérateurs, c'est à dire être déréférençables à l'aide de l'opérateur *, tout comme les pointeurs du C. Par conséquent mais indirectement, ceci implique l'existence d'itérateurs sur les eléments (de type `*Eit`). Ils seront naturellement déclarés dans la classe correspondante.

1.3.4 Bipoints

Cette entité est destinée à représenter comme son nom l'indique, un bipoint, c'est à dire un segment reliant deux nœuds. Ceci est dans le but de mesurer la longueur dans une métrique et de permettre la génération de maillages par bissection. La déclaration est très similaire à celle des faces topologiques :

```
template<class Vit,class Eit> class Bipoint
                                    : public Topology_Entity<Vit,Eit> ;
```

1.3.4.1 Contraintes sur les paramètres de template

Les contraintes sont les mêmes que pour les faces topologiques, §1.3.3.1. De plus , ils doivent contenir le membre `value_type` qui est un membre standard des itérateurs de la STL et qui renseigne sur le type de l'objet pointé. Il est parfaitement possible de se passer des itérateurs de la STL et d'utiliser des pointeurs « intelligents » à la place, mais il faudra penser à spécifier ce membre dans la classe support de ces pointeurs.

1.3.5 La classe contenant le maillage

1.3.5.1 Introduction

Pour le moment, cette entité est construite pour contenir un maillage uniforme de simplexes. Elle prend en paramètre la dimension de l'espace d'accueil, le nombre de nœuds des simplexes, et éventuellement un type de simplexe « usager », dans la déclaration suivante :

```
template<int n,int d,class E=Simplex<n,d,Vertex<d> > >
                                        class Simple_Mesh : public Mesh ;
```

La classe mère `Mesh` ne contient presque rien dans ce cas ci, et n'est pas de valeur explicative particulière. Au sein de la classe `Simple_Mesh` sont instanciés les containers qui servent a stocker toutes les entités utilisées (décrites plus haut). Cette classe contient toutes sortes de méthodes pour ajouter des nœuds, des éléments, faire du raffinement de maillage...etc. Il est à noter qu'elle fonctionne de façon très similaire à un container de la STL, puisque toutes les opérations sur des éléments internes se font grâce à des itérateurs. Ces itérateurs sont nécessaires car ils « informent » en quelque sorte l'usager externe de la façon avec laquelle les données sont stockées en interne. On pourrait penser que cette philosophie est en contradiction avec la POO (voir §1.1), mais il n'en est rien car ces itérateurs ne sont en fait que des indicateurs sémantiques, à l'instar des itérateurs de la STL. Ils ne servent donc qu'à guider le compilateur et à lui permettre d'optimiser plus efficacement les accès aux containers. (exemple : on peut retirer directement un élément d'un container connaissant un itérateur, mais pas si l'on ne dispose que d'un simple pointeur C). Dans les paragraphes précédents, il est mentionné quelques contraintes sur les paramètres de template, en particulier sur la notion d'itérateur. En fait ce n'est pas tout, car les itérateurs étant en principe associés aux containers, il est nécessaire de définir le container pour pouvoir définir l'itérateur.

En conséquence, les diverses entités devant être stockée dans cette classe devront fournir cette information aussi.

1.3.5.2 Containers

Chaque type d'entité « apporte » avec lui la façon avec laquelle les entités de ce type vont être stockées. Cela se matérialise par la présence d'un membre `Container_Type` déclaré public dans le corps de la classe concernée (nœud, élément, faces topologiques et bipoints). Typiquement, et tel que cela a été implémenté dans la librairie actuelle, il s'agit d'une liste chaînée, permettant les ajouts et les retraits rapidement, et surtout sans invalider aucun autre itérateur que celui pointant sur l'objet retiré, le cas échéant. Cette dernière propriété est très importante car, par exemple, un élément est construit directement à partir des itérateurs sur les nœuds le composant (voir §1.3.2). Voici un exemple de déclaration d'un tel container pour les éléments :

```
template<int n,int d,class E=Simplex<n,d,Vertex<d> > >
                                          class Simple_Mesh : public Mesh
{
public :
/// type d'elements (synonyme)
      typedef E Element_Type;
. . .

/// Iterateur vers un element (type synonyme)
      typedef typename Element_Type::Iterator Element_Iterator;
. . .
private :
/// containers (type synonyme)
      typedef typename Element_Type::Container_Type
                                                Element_Container_Type;
/// Instance du container
      Element_Container_Type Elements;
. . .
}
```

Par la suite, il suffit d'accéder au membre container nommé `Elements` comme avec n'importe quel container de la STL. De plus, les méthodes de la classe `Simple_Mesh`

communiqueront avec l'extérieur avec une philosophie proche de celle de la STL, qui est de ce point de vue, remarquable. Bien entendu, la classe `Simple_Mesh` est bien plus complexe qu'un simple container car elle doit contenir 4 type d'entités distinctes. Ceci a quelques conséquences sur les fonctions membres d'insertion par exemple, qui ne peuvent être nommées simplement « `insert` » ou « `push_front` » comme dans la STL. Ceci n'est pas le propos de cette explication ; le lecteur peut se référer au site internet décrivant la librairie pour des détails sur l'implémentation actuelle.

1.3.5.3 Classifications

Les structures de données (« containers ») de la classe de maillage ne sont pas supposées être performantes pour autre chose que pour faire du stockage d'information. Or, il est bien souvent nécessaire d'avoir recours à des recherches selon divers critères, comme on peut l'imaginer dans une base de données. Pour ce faire, il est nécessaire de construire des structures de données internes capables d'effectuer ces recherches de façon efficace. La recherche d'entités peut être menée pour plusieurs raisons :

- Imposer l'unicité des nœuds, auquel cas il faut faire une recherche avant l'insertion d'un nouveau nœud.
- Classer les bipoints selon leur longueur (pour faire du remaillage et ne couper que ceux qui sont plus longs qu'un certain critère).
- Connaissant un élément, trouver les faces topologiques le concernant (on les construit à partir des nœuds de l'élément).

La classification de chaque type d'entité est indépendante, et tous n'en ont pas besoin :

- Pour les nœuds ; classement lexicographique des coordonnées.
- Les éléments ne sont pas classés (juste ajoutés un à un dans leur container)

- **Les faces topologiques sont classées uniquement en fonction des nœuds les composant (connaissant ses nœuds, il est alors facile de retrouver une face)**
- **Les bipoints doivent être classés selon les mêmes critères que les faces (selon les nœuds), mais en plus il faut les classer par longueur pour les algorithmes de raffinement.**

Puisque chaque type d'entité contient déjà l'information sur le container l'accueillant (voir § pour une implémentation possible), il semble logique d'inclure l'information sur la ou les classifications associées. Par conséquent, les déclarations supplémentaires dans la classe `Simple_Mesh` ressembleront à celles exposées au §1.3.5.2, et se présentent ainsi (présenté uniquement pour les nœuds pour plus de clarté) :

```
template<int n,int d,class E=Simplex<n,d,Vertex<d> > >
                                          class Simple_Mesh : public Mesh
{
public :
/// type d'elements (synonyme)
      typedef E Element_Type;
/// type de noeud (synonyme)
      typedef typename Element_Type::Vertex_Type Vertex_Type;
/// Iterateur sur un noeud
      typedef typename Vertex_Type::Iterator Vertex_Iterator;
. . .
private :
/// type de containers pour noeuds
      typedef typename Vertex_Type::Container_Type
                                                  Vertex_Container_Type;
/// type de container pour la classification
      typedef typename Vertex_Type::Classification_Type
                                              Vertex_Classification_Type;
/// iterateur dans le container sur la classification
      typedef typename Vertex_Type::Classification_Iterator
                                          Vertex_Classification_Iterator;
. . .
/// Instance du container pour les noeuds
      Vertex_Container_Type Vertices;
/// Instance de la classification
      Vertex_Classification_Type Classification_Vertices;
. . .
}
```

Il est à noter que la classification détaillée ici reste de nature interne à la classe `Simple_Mesh` (mode d'accès « `private` »). De plus, la classification stocke des itérateurs, et non les objets directement (qui sont stockés dans un autre container).

1.3.5.4 Contraintes sur les types inclus dans la classe de maillage et les paramètres de template

Le paramètre `d` est un entier strictement positif. `n` est supérieur à 2 et inférieur à `d+1`. La classe `E` (ou son synonyme `Element_Type`) est une classe représentant un élément. Elle est optionnelle ; si elle est omise, une classe par défaut est utilisée. Dans le cas où elle est spécifiée, elle doit contenir l'information sur :

- Le type de nœud sur lequel l'élément est construit (`E::Vertex_Type`)
- Le type de face topologique associé à l'élément (`E::Topological_Face_Type`)
- Le type de bipoint associé à l'élément (`E::Bipoint_Type`)
- Le type de container utilisé pour le stockage des éléments (`E::Container_Type`)
- Le type d'iterateur associé au container (membre `E::Iterator`)
- Éventuellement une classification (non implémentée pour le moment)

En addition, le type `Vertex_Type` déclaré dans la classe E doit impérativement fournir les informations suivantes :

- Le type de container utilisé pour le stockage des nœuds (`Vertex_Type::Container_Type`)
- Le type d'itérateur de ce container (`Vertex_Type::Iterator`), imposé en outre au §1.3.2.1.1.

- **Le type du container de classification**
  (`Vertex_Type::Classification_Type`)
- **Le type d'itérateur associé a la classification**
  (`Vertex_Type::Classification_Iterator`)

Le type `Topological_Face_Type` **doit fournir quand à lui :**

- **Le type de container utilisé pour le stockage des faces**
  (`Topological_Face_Type::Container_Type`)
- **Le type d'itérateur pour ce container**
  (`Topological_Face_Type::Iterator`)
- **Le type du container de classification par les nœuds (pour recherche rapide et éviter les doublons)**
  (`Topological_Face_Type::Classification_Type`)
- **Le type d'itérateur associé a la classification**
  (`Topological_Face_Type::Classification_Iterator`)

Le type `Bipoint_Type` **doit fournir :**

- **Le type de container utilisé pour le stockage des faces**
  (`Bipoint_Type::Container_Type`)
- **Le type d'itérateur pour ce container**
  (`Bipoint_Type::Iterator`)
- **Le type du container de classification par les nœuds (pour recherche rapide et éviter les doublons)**
  (`Bipoint_Type::Classification_A_Type`)
- **Le type d'itérateur associé a cette la classification**
  (`Topological_Face_Type::Classification_A_Iterator`)

- **Le type du container de classification par longueur**

  (Bipoint_Type::Classification_B_Type)

- **Le type d'itérateur associé a cette la classification**

  (Topological_Face_Type::Classification_B_Iterator)

Ces contraintes se matérialisent dans les classes correspondantes de la façon suivante (exemple de classe de nœud) :

```
template<int N> class Vertex : public Vertex_Base<N>
{
. . .
public :
. . .
/// type de container (ici une liste chainee)
      typedef list<Vertex<N> > Container_Type;
/// iterateur associe
      typedef typename Container_Type::iterator Iterator;
/// type de classification (set)
      typedef set<Iterator,Less_Ptr_Type<Iterator> >
                                                  Classification_Type;
/// iterateur associe a la classification
      typedef typename Classification_Type::iterator
                                                Classification_Iterator;
. . .
}
```

Dans l'exemple ci dessus, Less_Ptr_Type<Iterator> est un des « functor » qui permet de comparer les objets pointés par les itérateurs stockés dans le container de classification. Il est hors du propos de cette introduction de les décrire en détail ; pour une référence plus complète, le lecteur est invité à consulter la documentation de la STL [80] ou celle du code de la libraire de maillage [81].

En conclusion, cela peut sembler beaucoup de contraintes, mais il s'agit bien du minimum. Une version antérieure de la librairie en comportait bien plus ! Toutes ces contraintes correspondent à une déclaration dans la classe `simple_mesh`. Il serait trop fastidieux de les présenter toutes ici, de plus cela n'est pas d'une grande puissance

explicative. Un exemple type pour les nœuds existe dans les paragraphes §1.3.5.3 et §1.3.5.2.

1.4 Utilisation de la classe de maillage

Ce document n'a pas l'intention d'être un tutorial de la libraire, mais il est bon de présenter un exemple d'utilisation du code tel que présenté montrant la similitude qui peut exister avec la STL. Dans l'exemple qui suit, on insère successivement quelques triangles dans un maillage et l'on effectue un raffinement isotrope de la surface ainsi constituée.

```
      // déclarations :
      // champ de metrique (ici taille constante)
Constant_Isotropic_Metric<Vertex<2> > CIField(0.05);

      /* un maillage utilisant la metrique et utilisant le type
      d'élément par défaut (3 noeuds=triangle, et dimension=2)*/
Simple_Mesh<3,2> M(CIField);

      // un noeud de ce maillage
Simple_Mesh<3,2>::Vertex_Type V;
      // un element de ce maillage
Simple_Mesh<3,2>::Element_Type E;
      // des iterateur sur des noeuds du maillage
Simple_Mesh<3,2>::Vertex_Iterator VI[4];
      // des iterateur sur des elements du maillage
Simple_Mesh<3,2>::Element_Iterator EI[2];

      // création des noeuds
V[0]=0.; V[1]=0.;                    // premier noeud
VI[0]=M.Add_Vertex(V);               // stocké dans le maillage
V[0]=1.; V[1]=0.;                    // second noeud
VI[1]=M.Add_Vertex(V);
V[0]=0.;    V[1]=1.;                 // troisieme noeud
VI[2]=M.Add_Vertex(V);
V[0]=1.;    V[1]=1.;                 // quatrieme noeud
VI[3]=M.Add_Vertex(V);

      // création des éléments
E[0]=VI[0]; E[1]=VI[1]; E[2]=VI[2];
EI[0]=M.Add_Element(E);              // 1er element dans le maillage
E[0]=VI[1]; E[1]=VI[2]; E[2]=VI[3];
EI[1]=M.Add_Element(E);              // 2nd element dans le maillage
```

```
      // création des connectivités
M.Attach(EI[0]);
M.Attach(EI[1]);
M.Save_PLT_File("init.plt");        // sauvegarde du maillage
                                    // initial
      // maillage
M.Do_Mesh(1.3,0.0);                 // on fait le maillage
M.Save_PLT_File("rest.plt");        // sauvegarde du maillage
                                    // final
```

Le résultat de ce programme est donné dans la figure suivante :

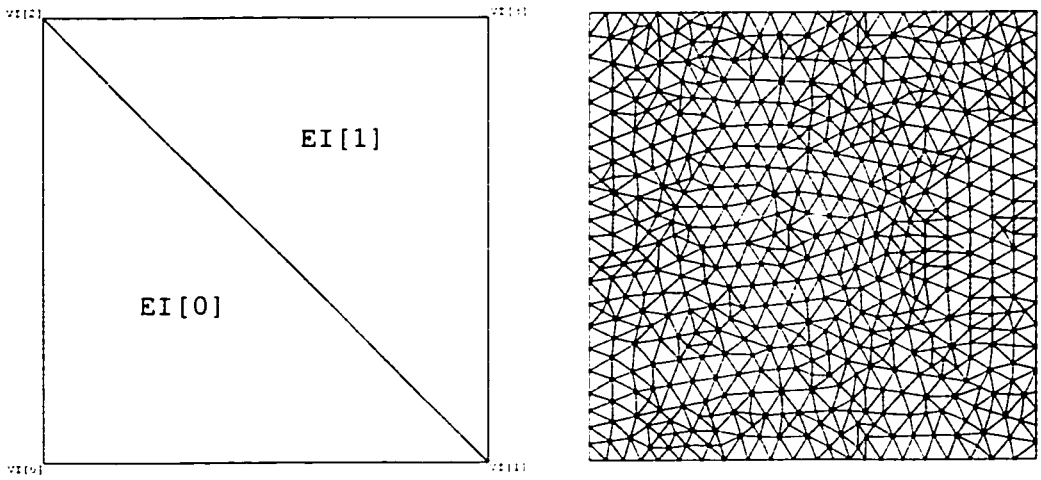


Figure 1 : Maillage d'origine et maillage résultat

1.5 Conclusion

Cet brève introduction avait pour but de montrer l'originalité de la structure de la librairie de maillage. Cette structure de programmation générique est un paradigme fondamentalement différent de celui de la programmation objet généralement acceptée. Le propos de ce chapitre n'est pas de montrer que cette dernière n'est pas adaptée, bien au contraire. On peut d'ailleurs comprendre la programmation par templates comme une forme cachée d'héritage statique. En effet, on impose aux paramètres de template des

classes décrites ici des contraintes d'ordre sémantique (i.e. telle classe doit contenir tel membre ou telle fonction membre pour s'interfaçer avec la classe de maillage). Ces contraintes sont exactement du même ordre que celles que l'on obtient lorsque l'on fait dériver une classe B d'une autre classe A: on doit coder un certain nombre de méthodes de la classe B déjà présentes dans la classe A. Ceci est obligatoire pour rendre la classe B compatible avec l'interface (commune) fournie par la classe A. Cette dernière possède d'ailleurs en général des méthodes virtuelles pures que l'on doit coder de toute façon. On peut se demander où se trouve la différence conceptuelle entre une programmation orientée objet classique et cet exercice de programmation générique ; puisque en suivant ce raisonnement, les deux approches (suivant des sémantiques fort différentes) en arrivent au même point. Il y a en fait une différence de taille : là ou en programmation orientée objet classique le type des objets manipulés n'est connu qu'a l'exécution, dans le cadre de la programmation par template, il est impérativement connu à la compilation. Ceci a deux conséquences : premièrement, la programmation objet classique est plus souple en termes d'utilisation (du fait de la présence de méthodes virtuelle et du mécanisme d'héritage). Deuxièmement, la programmation par template est la plus performante en termes de vitesse d'exécution (car le compilateur connait tout des objets qu'il manipule : il est ainsi capable d'utiliser des mécanismes d'optimisation qui autrement ne seraient pas efficaces). La morale de cette conclusion est la suivante : pour de grosses application ne nécessitant pas de performance numériques, il est plus facile de programmer en utilisant le paradigme orienté objet. Pour des applications au contraire sensibles a la vitesse de calcul, mieux vaut s'orienter dans la programmation générique par template (dans la mesure du possible !)

1.6 Références

[77] American National Standards Institute (ANSI), 1998, ISO/IEC 14882-1998 « Information Technology - Programming Languages - C++ », ANSI.

[78] M. A. Ellis, B. Stroustrup, 1991, The Annotated C++ Reference Manual, Addison-Wesley, ISBN 0-201-51459-1.

[79] David R. Musser, Gillmer J. Derge, and Atul Saini, 2001, STL Tutorial and Reference Guide, Second Edition: C++ Programming with the Standard Template Library, Addison-Wesley, ISBN 0-201-37923-6.

[80] SGI inc, Standard Template Library Programmer's Guide, site web : http://www.sgi.com/tech/stl

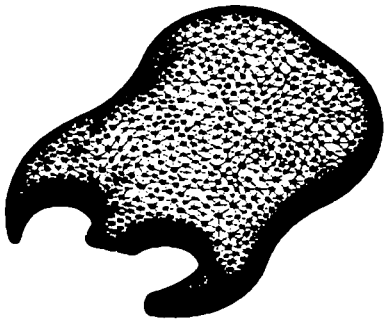
[81] Eric Béchet, 2002, documentation de la librairie de maillage, site web : http://www.bechet.ca.tc/doc

ANNEXE 2

DIAPOSITIVES DE LA PRÉSENTATION

Résolution d'un problème aux limites à frontière libre au moyen d'un algorithme de remaillage adaptatif et anisotrope

Éric Béchet
École Polytechnique de Montréal
Université du Québec à Trois-Rivières



Plan

1- Introduction

2- Analyse des besoins

3- Génération de maillage

4- Évolution du front de matière

5- Simulations isothermes

6- Simulations non-isothermes

7- Autres stratégies d'adaptation

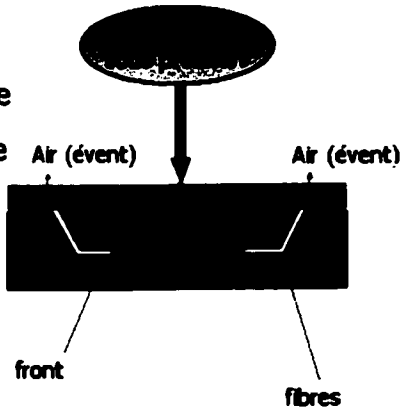
8- Conclusion

9- Recommandations

1- Introduction

Procédé RTM

- Renfort fibreux placé dans le moule
- Fermeture et verrouillage du moule
- Injection de résine sous pression



Objectifs de la simulation :

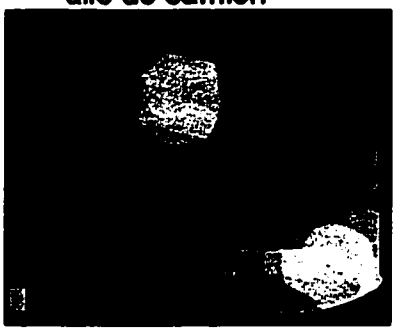
-> Éviter les zones sèches

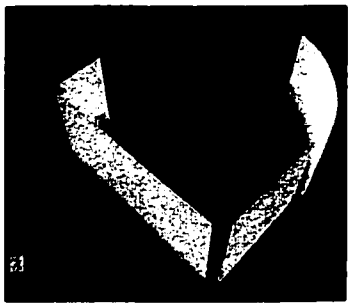
-> Optimiser le temps de cycle

Exemples

Pièces aéronautiques

Pièce automobile : aile de camion





Progression du front de résine dans le temps

Modélisation

$$\vec{V} = \frac{-\overline{\overline{K}}}{\mu}.\vec{\nabla P}$$

Équation de Darcy (2D ou 3D)

milieu poreux

$$u = \frac{K_x}{\mu}\left(-\frac{\partial P}{\partial X}\right)$$

Equation de Darcy (en 1D)

u : vitesse dans la direction X.

K_x : Permeabilité du milieu poreux

μ : Viscosité de la résine.

$\frac{\partial P}{\partial X}$: Gradient de pression (dir. X)

2- Analyse des besoins

Simulations actuelles :

- Maillage fixe

- Progression du front de résine par « contact »

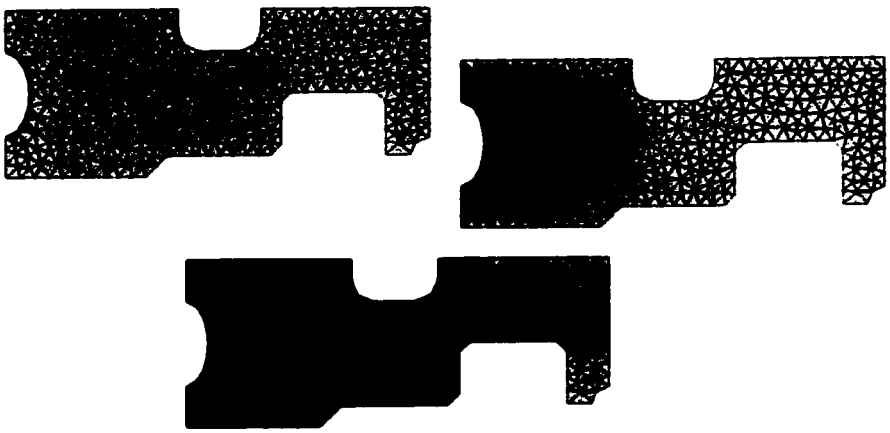
Conséquences :

-Aucun contrôle sur le nombre de pas de temps d'une simulation, et donc sur la précision (dépend directement du maillage)

-Temps calcul de l'ordre de n^3 , n étant le nombre de ddl.

-Conditions aux limites mal approchées (au front)

Simulations actuelles



Besoins

- Possibilité de jouer indépendamment sur la discrétisation temporelle et la discrétisation spatiale
- Tendre vers un temps de calcul d'ordre inférieur
- Améliorer l'imposition des conditions aux limites
- Adapter localement la discrétisation pour tenir compte des:
    - > effets de bord
    - > fronts thermiques

Solution envisagée

a) Utilisation de méthodes de remaillage sur coques 3D
   - adaptatives
   - anisotropes

b) Couplage avec un méthode d'évolution de font par surfaces de niveau (Level Set)

c) Application au RTM
   - isotherme, non isotherme

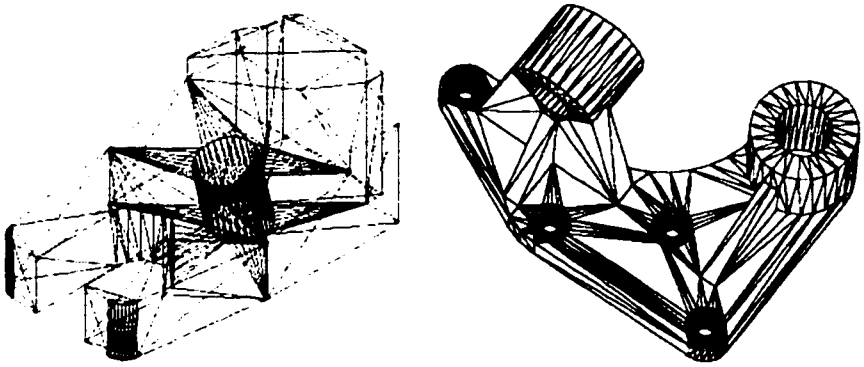
d) Génération de maillage initial optimisé *a priori*

3- Génération de maillage

- Maillage surfacique pour coques 3D
- Extensions anisotropes
- Construction d'une métrique

Base géométrique

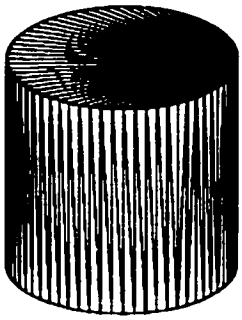
- On peut partir d'un fichier STL



Base géométrique

- Représentation géométrique inexacte
- Les triangles sont étirés arbitrairement selon les courbures de la surface
+ Ne dépend pas du système CAO
+ Peut approcher la surface avec tout niveau de précision requis

→ Ne peut pas être utilisé avec les éléments finis



Bissection

- Les nouveaux nœuds sont générés par bissection des segment existant (par ordre de longueur)
- La localisation du nœud repose sur le rapport taille courante / taille voulue :

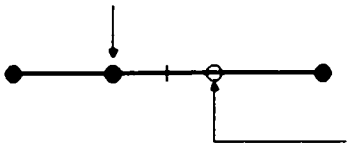
$$\alpha_i = \text{int}\left(\frac{d_{actual}}{d_{target}}\right)$$

$$\begin{cases} \gamma = \frac{1}{2} - \frac{1}{2 \cdot \alpha_i} & \text{if } \alpha_i \text{ is odd} \\ \gamma = 0.5 & \text{else} \end{cases}$$

Bissection

Etape 1: $d=3*d_{target}$; $\gamma=1/3$

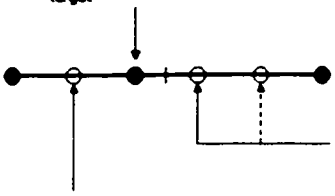
Exemple 1



Etape 2: $d=2*d_{target}$; $\gamma=1/2$

Etape 1: $d=5*d_{target}$; $\gamma=4/10$

Exemple 2

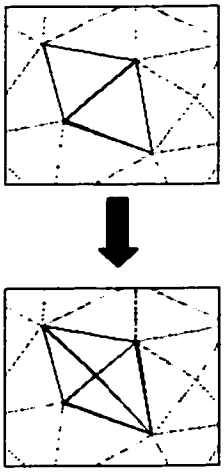


Etapes 2: $d=3*d_{target}$; $\gamma=1/3$ (schéma connu)

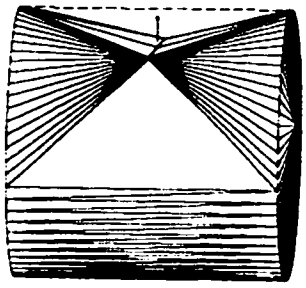
Etape 3: $d=2*d_{target}$; $\gamma=1/2$ (schéma connu)

Etc..

Projection

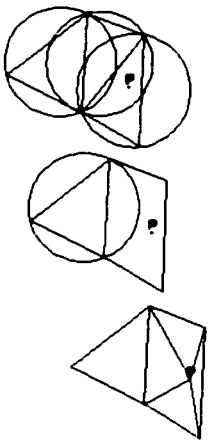


- Le noeud doit être projeté sur la surface d'origine



Remaillage Delaunay

- Nous utilisons le critère de Delaunay (du cercle circonscrit vide), en 2D.
- Nous remaillons le voisinage du noeud inséré à l'aide d'un algorithme de "De Bowyer-Watson" et de retournements d'arêtes pour conserver localement le critère de Delaunay



Comme nous travaillons sur des surfaces courbes, il faut:

a) Changer le critère du cercle vide

b) Éviter les dégénérescences géométriques

Nouveau critère de Delaunay

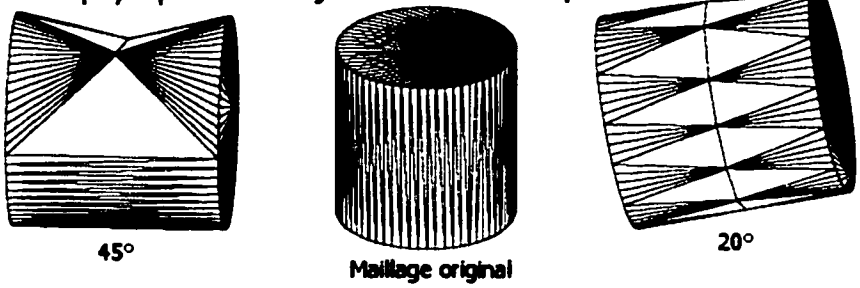
- Au lieu du cercle, nous utilisons la sphère équatoriale.
  - Chaque noeud du triangle est sur la sphère et son centre appartient au plan formé par le triangle.
- Le même algorithme de remaillage est utilisé par la suite

Nouveau critère de Delaunay

Delaunay-conformité étendue aux domaines...

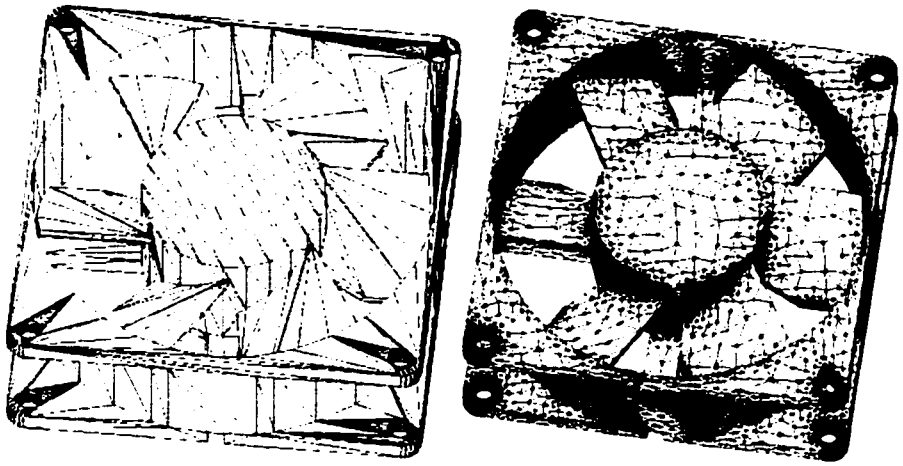
Géométrie conservée

- Nous utilisons un angle limite pour les retournements de segments dans l'algorithme de "De Bowyer-Watson" Une bonne valeur est 20°.
- Évidemment, il est interdit de retourner les arêtes physiques de l'objet déterminées a priori

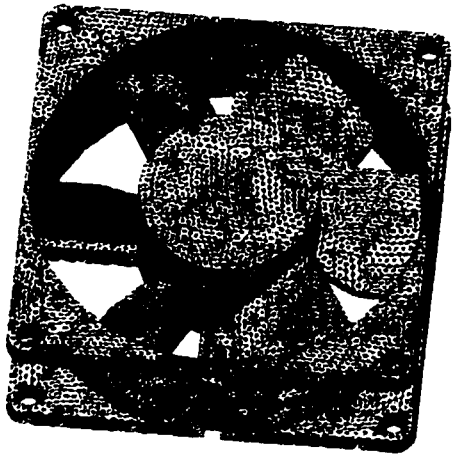


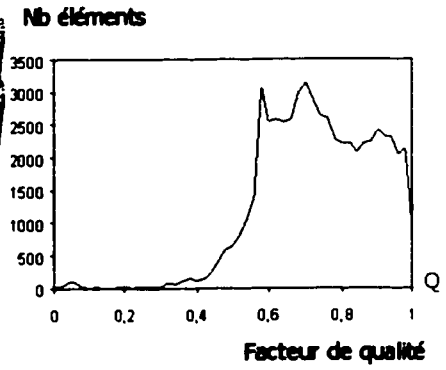
45° Maillage original 20°

Exemples - Ventilateur

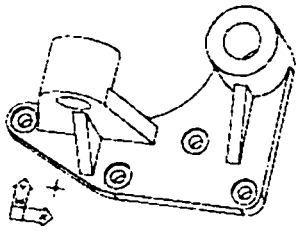


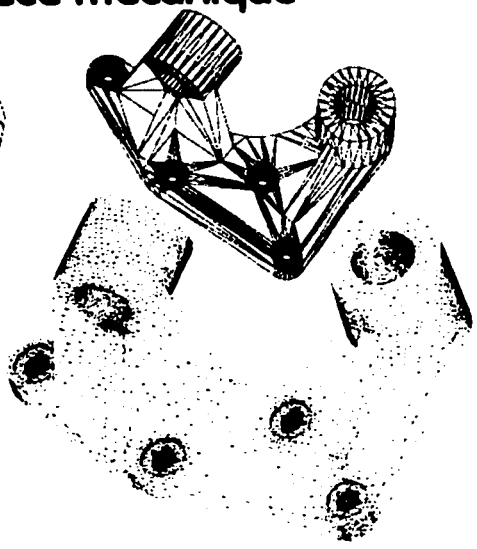
Exemples - Ventilateur





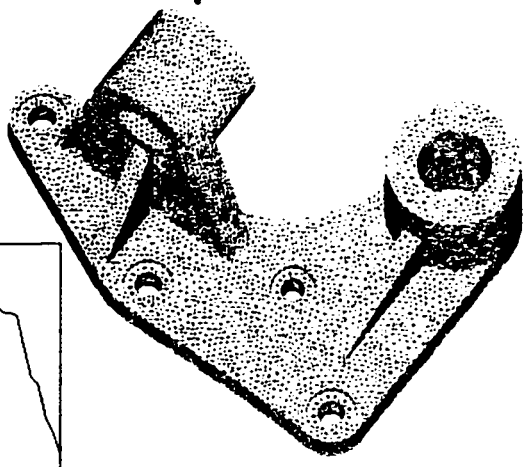
Exemples – Pièce mécanique

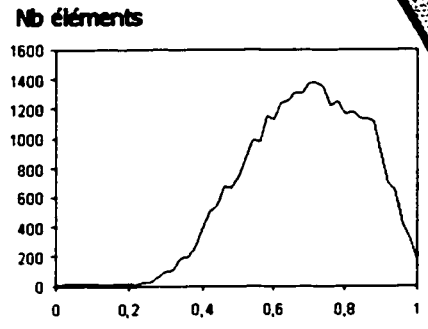




Source : Pr. V. Francois,
Dept. Génie Mécanique,
UQTR, Trois Rivières,
Québec.

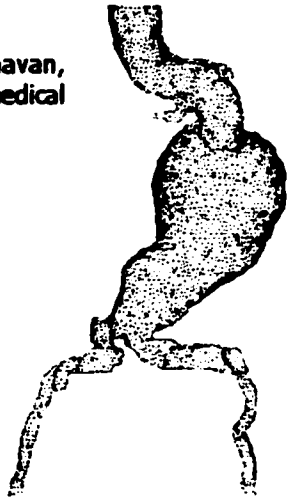
Exemples – Pièce mécanique



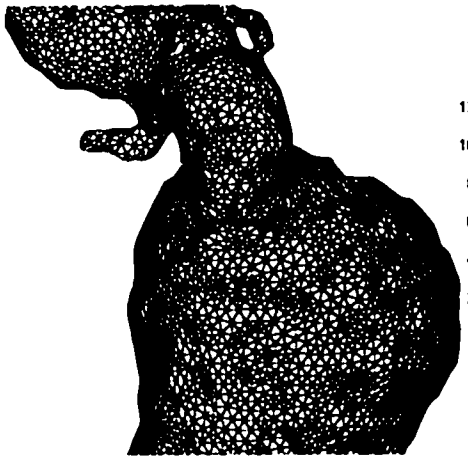


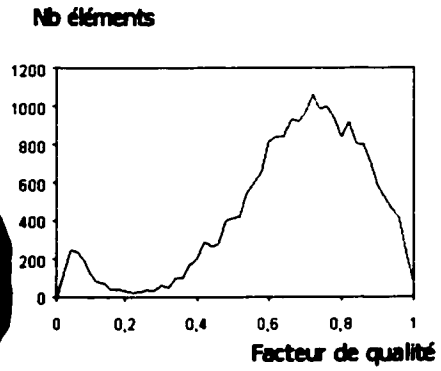
Exemples – Anévrisme

Source : Dr. M.L. Raghavan,
Department of Biomedical
Engineering,
University of Iowa
Iowa City, IA



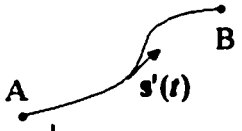
Exemples – Anévrisme





Anisotropie : nouvelle notion de distance

$$\mathbf{M}(P)=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}$$



$$dist(AB)=l(\Gamma)=\int_0^1 \sqrt{{}^t\mathbf{s}'(t)\cdot\mathbf{M}\left(s(t)\right)\cdot\mathbf{s}'(t)}dt$$

- Nous calculons la longueur du segment par un schéma d'intégration de Simpson-
- ou utilisons la quadrature suivante si la métrique **M** est monotone le long de *AB* :

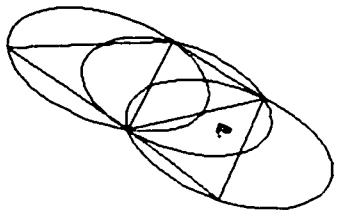
$$dist(AB)=\frac{\sqrt{{}^t\overrightarrow{AB}\cdot\mathbf{M}(A)\cdot\overrightarrow{AB}}+\sqrt{{}^t\overrightarrow{AB}\cdot\mathbf{M}(B)\cdot\overrightarrow{AB}}}{2}$$

Nouveau critère de Delaunay (2)

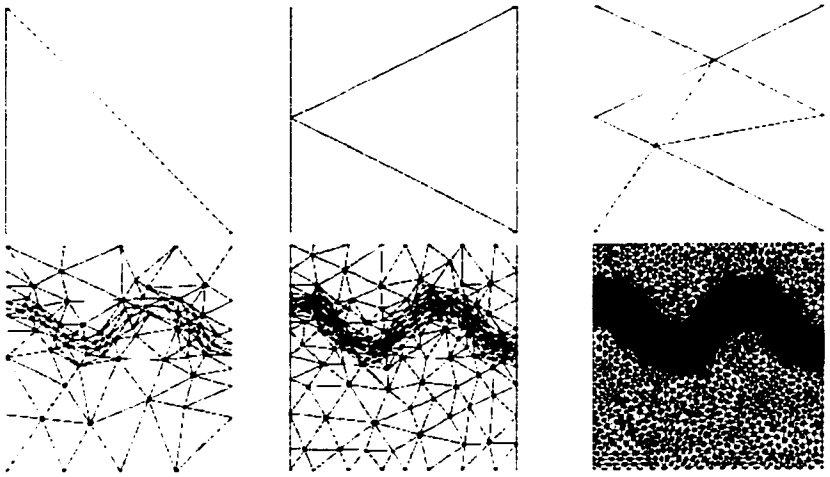
Critère de la sphère vide -> Maillages isotropes

La notion de métrique « distord » l'espace, les sphères deviennent donc des ellipsoïdes.

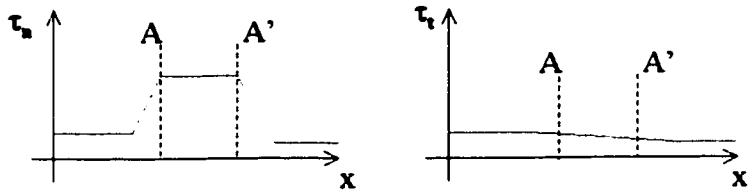
Le critère modifié est donc un critère basé sur les ellipsoïdes vides (dans l'espace réel).



Maillages anisotropes "partiels"



Définition de la métrique



Définition de la métrique

Densités normales et tangentielles: sont utilisées pour déterminer la métrique dans un base locale

$$s_n = \frac{1}{\tau_n} = \frac{\vec{\mathbf{V}} \cdot \vec{\mathbf{n}}}{\Delta t \cdot k} \qquad s_t = \frac{1}{\tau_t} = cst$$

Ces densités sont valides au voisinage d'épaisseur *e* du front:

$$e = (AA') = 3 \cdot \vec{\mathbf{V}} \cdot \vec{\mathbf{n}} \cdot \Delta t$$

Rotation de la métrique

Définition de la métrique et transfert dans la base globale (cas 2D):

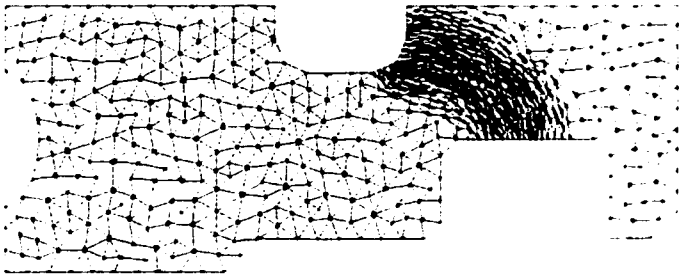
$$\mathbf{M} = {}^{\mathrm{T}}\mathbf{R} \cdot \begin{bmatrix} \tau_n^{-2} & 0 \\ 0 & \tau_t^{-2} \end{bmatrix} \cdot \mathbf{R}$$

Matrice de rotation :

$$\mathbf{R} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \qquad \theta = \left(\vec{Ox}, \vec{\mathbf{n}}\right)$$

Génération du maillage

Le nouveau maillage est généré en utilisant la métrique, à l'aide des algorithmes de bissection.



4- Évolution du front

- Surfaces de niveau (Level-Sets)
- Transport de la vitesse dans le domaine sec
- Génération d'un champ de distance *d(x,y,z)* / front
- Génération d'un champ de temps
- Mise à jour du front

Surfaces de niveau (Level Sets)

Méthode générale d'évolution de front

$$\frac{\partial u}{\partial t} + v(\mathbf{X},t)\cdot|\nabla u| = 0 \text{ in } \Omega \qquad \frac{\partial u}{\partial t} + \mathbf{V}(\mathbf{X},t)\cdot\nabla u = 0$$

$$u(\mathbf{X},t) = 0 \text{ on } \Gamma(t)$$

$$\Gamma(t_0) = \Gamma_0$$

- La vitesse normale v est quelconque
- Problème à valeur initiale
- Résolution par une méthode de transport; implémentation relativement lourde.

Surfaces de niveau (Level Sets)

RTM - évolution du front monotone

-> chaque point n'est « atteint » par le front qu'une fois.

-> utilisation d'une méthode simplifiée: équation eikonale

$$v(\mathbf{X})\cdot|\nabla u| = 1 \text{ in } \Omega \qquad u = g(\mathbf{X}) \text{ on } \Gamma$$

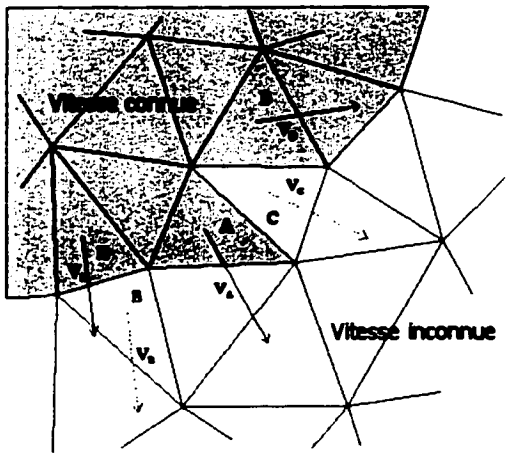
- La vitesse normale v est toujours positive
- Problème aux limites
- Résolution par « fast marching », implémentation plus légère (schéma « upwind »).

Transport du champ de vitesse

- Connu originellement uniquement dans la région saturée de résine (par simulation numérique), il est par conséquent nécessaire de l'extrapoler.
- Champ vectoriel, constant par éléments
- L'extrapolation se fait couche d'éléments par couche d'éléments, à partir du champ trouvé dans les éléments situés juste en arrière du front.

  -> tenir comte des perméabilités

Transport du champ de vitesse

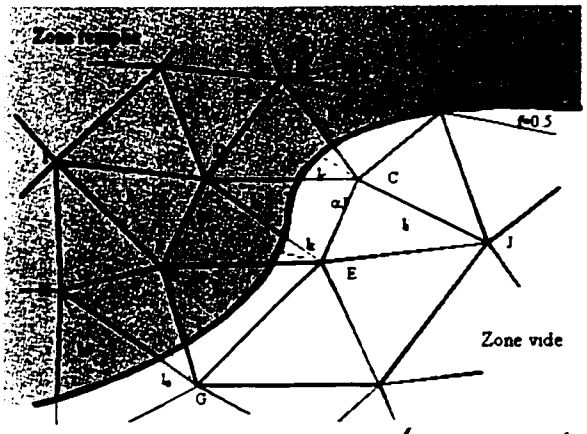


Génération d'un champ de distance

Il est construit couche par couche à partir des éléments traversés par le front , iso-valeur de remplissage =0.5. Cette iso-valeur est construite à partir du pas de temps précédent

- C'est un champ scalaire
- Continu et linéaire par élément
- Il représente la distance « topologique » entre un nœud du maillage et le front

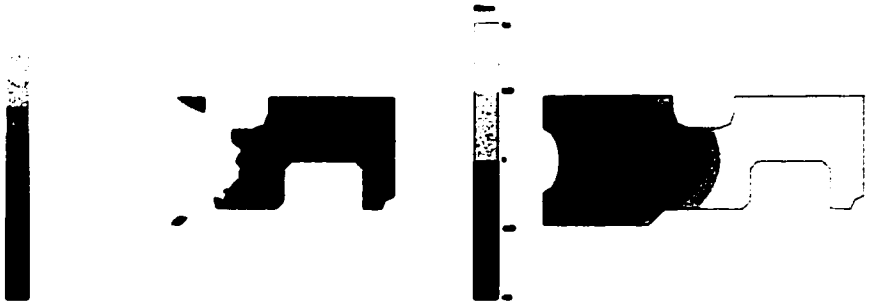
Génération d'un champ de distance



$$d_K = \left(\alpha_K \cdot d_{N_0} + \left(1-\alpha_K\right) \cdot d_{N_1}\right) + l_K$$

Génération d'un champ de distance

Le champ de distance permet de « s'y retrouver » par rapport au front de résine dans un maillage non structuré

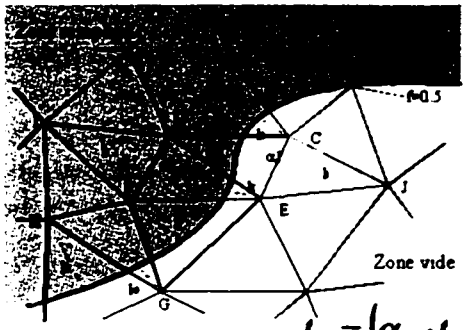


Génération du champ de temps

Il est construit couche par couche à partir des éléments traversés par le front, très similairement au champ de distance

- C'est un champ scalaire
- Continu et linéaire par élément
- Il représente une estimation pour chaque noeud du temps au bout duquel il sera « touché » par le front

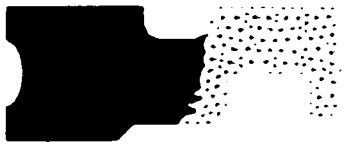
Génération du champ de temps



$$t_K = \left(\alpha_K \cdot t_{N_0} + (1-\alpha_K)\cdot t_{N_1}\right) + \Delta t_K$$

$$\Delta t_k = \frac{\|\overrightarrow{\nabla d}\|}{\vec{v}.\overrightarrow{\nabla d}} \cdot \left(d_k - \left(\alpha_K \cdot d_{N_0} + (1-\alpha_K)\cdot d_{N_1}\right)\right)$$

Génération du champ de temps

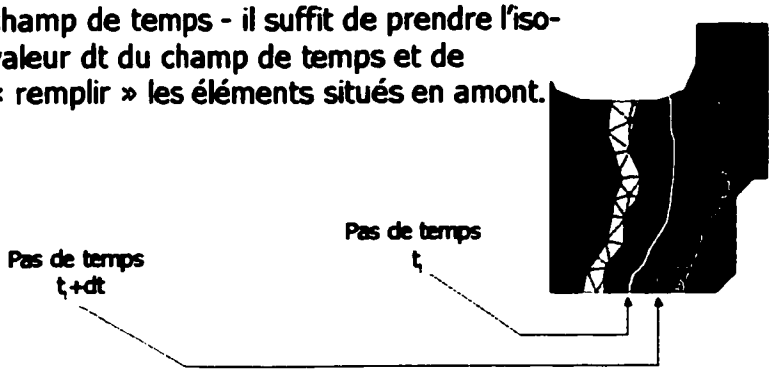


Facteur de remplissage

Champ de temps

Mise à jour du front

La position du front pour $t_{i+1}=t_i+dt$ est mise à jour sur le <u>nouveau maillage</u> en utilisant le champ de temps - il suffit de prendre l'iso-valeur dt du champ de temps et de « remplir » les éléments situés en amont.



5- Simulations isothermes

-Pas de phénomènes thermiques pris en compte

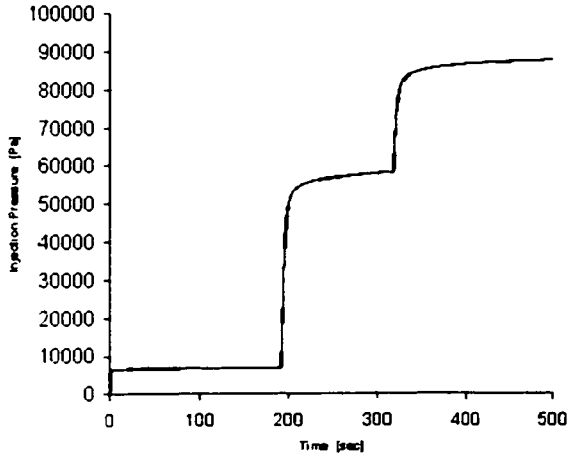
-> pas de phénomènes de transport

-> pas d'effets dynamiques

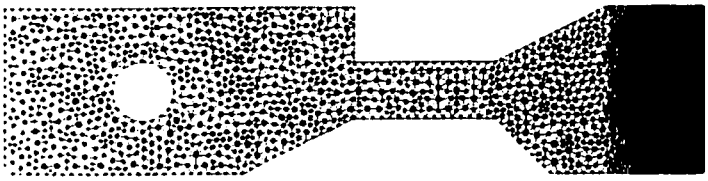
Les paramètres sont fondés sur une expérience réelle :

- Injection contrôlée en pression
- Tissu isotrope $K=5,2.10^{-10}\ m^2$
- Huile de silicone, viscosité dynamique $\mu=0,1\ N.s.m^{-2}$

Pression d'injection

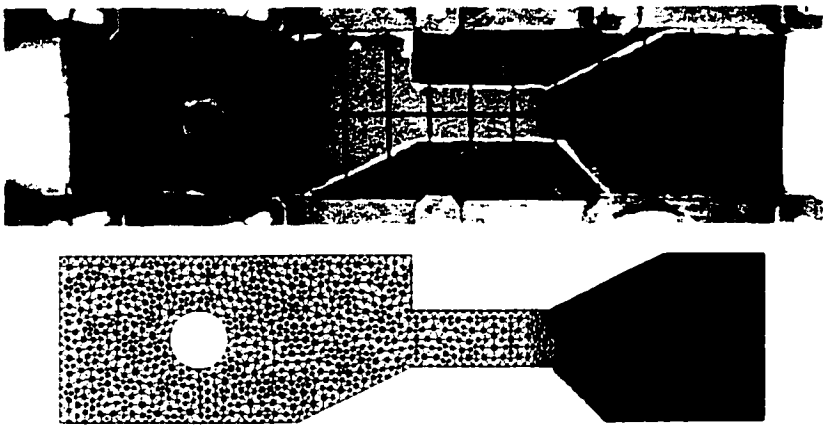


Résultats - comparaison



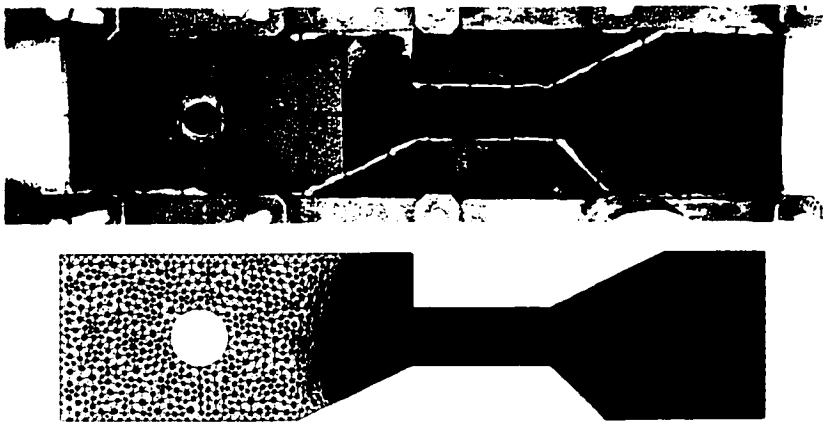
T=34 s

Résultats - comparaison



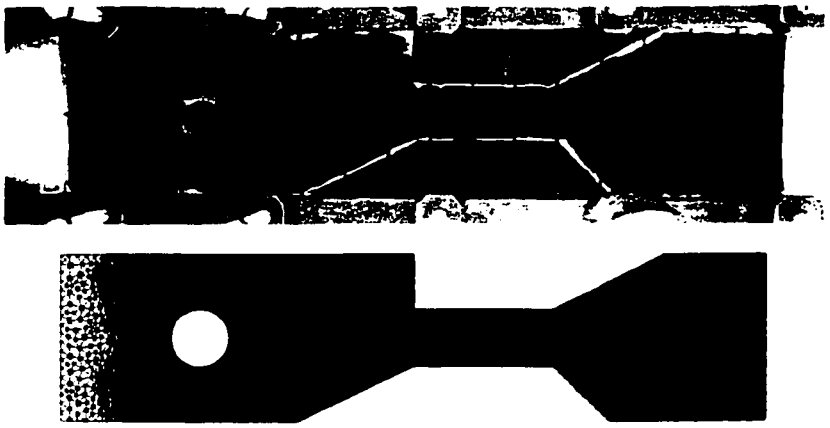
T=178 s

Résultats - comparaison



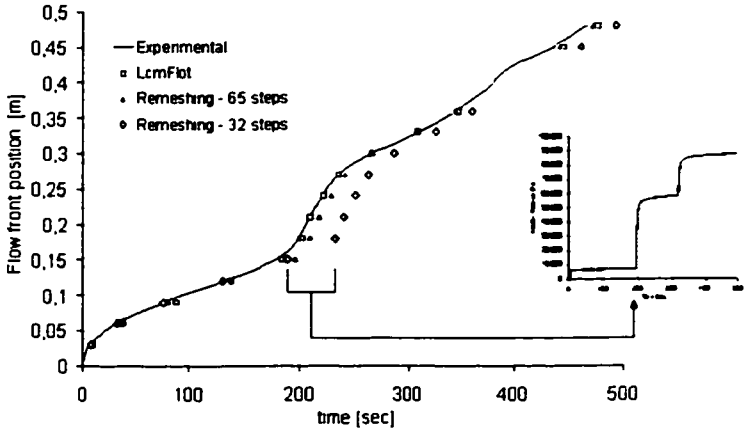
T=266 s

Résultats - comparaison

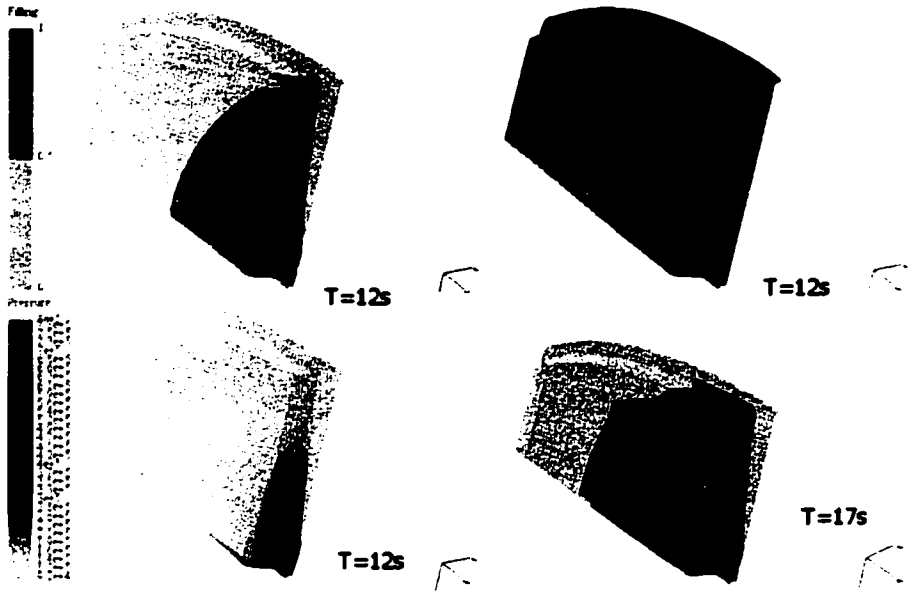


T=435 s

Résultats - comparaison



Autre Résultat



Analyse

- Forte diminution du nombre de pas de temps
- Front très lisse
- Meilleure imposition des conditions aux limites

mais

- Chaque pas de temps est plus lourd à calculer

(remaillage + résolution du pb de Darcy)

6- Simulations non isothermes

- Une résine de température diff. de celle du moule
- Polymérisation - exothermie
- Phénomènes de transport (espèces chimiques , température)

$$\left(\langle \rho c_p \rangle\right)\frac{\partial \langle T \rangle}{\partial t} + \rho_f c_{p_f} \langle \mathbf{v} \rangle \cdot \nabla \langle T \rangle =$$

$$\nabla\left(\mathbf{k}\nabla\langle T \rangle\right) - \rho_f \Delta H \frac{d\langle \chi \rangle}{dt} - \langle \mathbf{v} \rangle \nabla \langle p \rangle$$

Aspects particuliers

- Équation de la chaleur:
  - Diffusion (Galerkin std) . Implicite.
  - Transport (Lesaint-Raviart)
- Taux de polymérisation *
  - Intégration (Runge-Kutta o. 4)
  - Transport (Lesaint-Raviart)

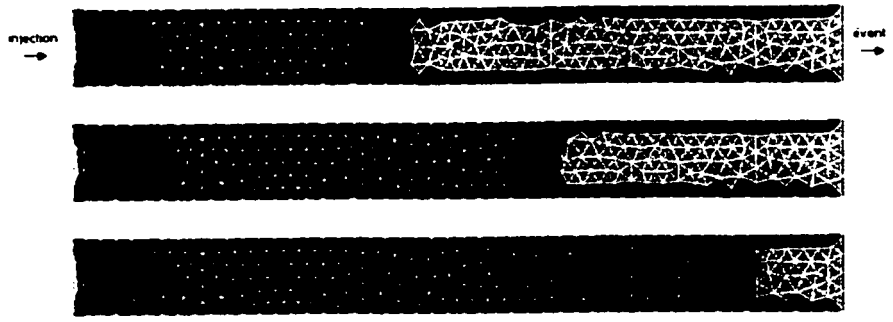
-> la temp. doit être conservée sur plusieurs pas de temps (t_{i-2},t_{i-1},t_i)

Simulations

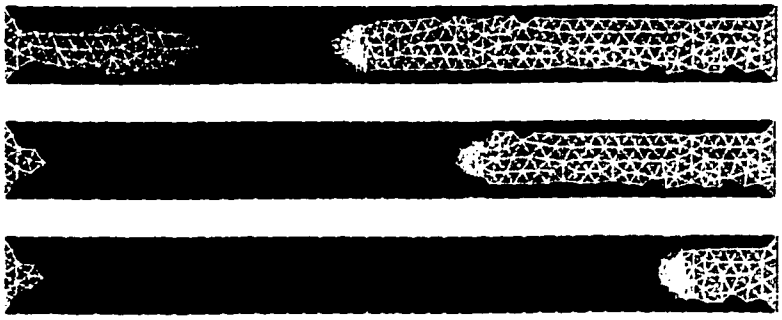
Température résine injectée : 340 K
Viscosité dynamique : 0.1 Pa.s
Température moule : 400 K
Température fibres : 340 K
Pression d'injection : 2.10^5 Pa
Pas de réaction-> pas d'exothermie
Cp fibres << Cp résine
$a = b = 2{,}01.10^6$ J.K^{-1}.m^{-3}
$k = 0.25$ J.m^{-1}.K^{-1}

$$a\frac{\partial T}{\partial t} + b \cdot \mathbf{v} \cdot \nabla T = \nabla(\mathbf{k} \cdot \nabla T)$$

Simulation classique



Simulation avec remaillage interactif



Problèmes

Unique changement : le pas de temps

Formulation purement eulerienne

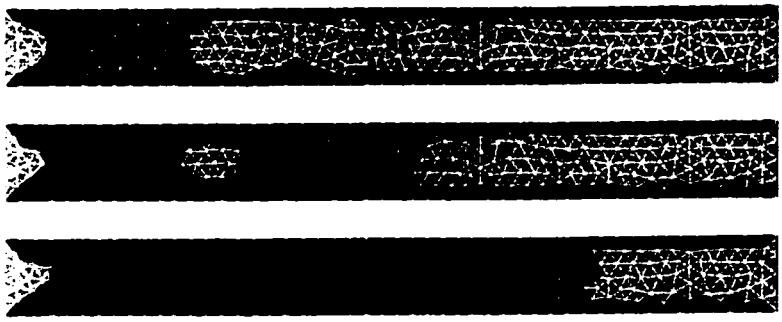
-> condition Courant-Friedrichs-Lewy (CFL) à respecter pour Lesaint-Raviart

- Transport de la température
- Transport du taux de polymérisation

La condition CFL impose que le nombre de Courant $C=(v.\Delta t)/\Delta x$ soit inférieur a une fraction de l'unité.

CFL=0.25 sans remaillage vs. CFL = 1.5 avec remaillage.

Simulation standard – grand pas de temps



Instable, CFL =1.5

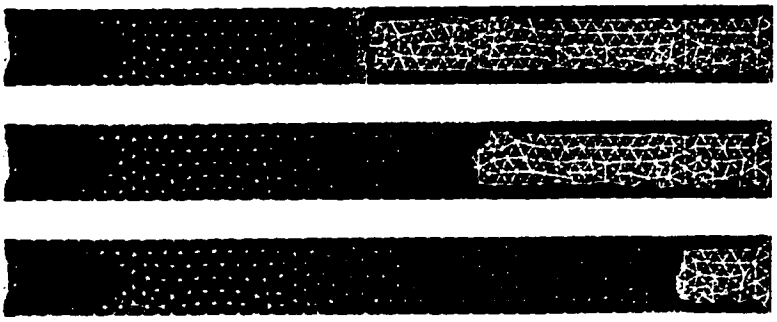
Pas de temps adaptatif

Stabilisation du schéma : pas de temps différents pour le problème de Darcy (évolution du front) et le problème thermique :

- Darcy : pas de condition sur le pas de temps (toutefois contraintes de précision)
- Thermique : respect obligatoire de la condition CFL

Cas présent : 3 pas de temps en thermique pour 1 pas de temps de Darcy.

Pas de temps adaptatif



Analyse

- Réduction de la diffusion artificielle au front
- Meilleure imposition des conditions aux limites thermiques au front
- Contrôle du pas de temps (on peut minimiser la diffusion numérique)

mais

- Chaque pas de temps est plus lourd à calculer
- Grand nombre de pas de temps thermiques (méthodes Euleriennes)

7- Autres stratégies d'adaptation

Maillage unique pour toute la simulation, généré *à priori*. La cavité est remplie, et un calcul est fait en régime permanent.

- Adaptation en erreur d'interpolation
- Adaptation par rapport au déplacement du front
- Adaptation en nombre de Courant (eq de transport)
- Comparaisons

Erreur d'interpolation

Estimateur d'erreur

Matrice hessienne $\mathbf{H}(p)=\begin{bmatrix} \frac{\partial^2 p}{\partial x^2} & \frac{\partial^2 p}{\partial y \partial x} & \frac{\partial^2 p}{\partial z \partial x} \\ \frac{\partial^2 p}{\partial x \partial y} & \frac{\partial^2 p}{\partial y^2} & \frac{\partial^2 p}{\partial z \partial y} \\ \frac{\partial^2 p}{\partial x \partial z} & \frac{\partial^2 p}{\partial y \partial z} & \frac{\partial^2 p}{\partial z^2} \end{bmatrix}$

Valeurs propres/vecteurs propres $\mathbf{H}={}^{\mathrm{T}}\mathbf{E}\cdot\mathbf{\Lambda}\cdot\mathbf{E}$

Métrique uniformisant l'erreur à la valeur ε:

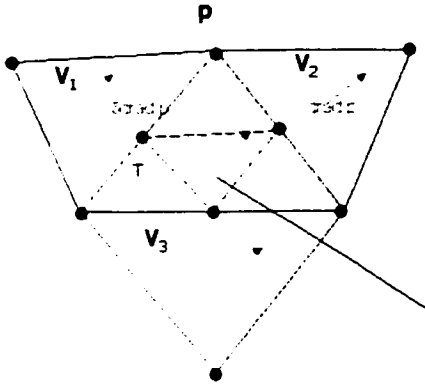
$$\mathbf{M}(p)={}^{\mathrm{T}}\mathbf{E}\cdot|\mathbf{\Lambda}|\cdot\mathbf{E}\cdot\frac{1}{\varepsilon}$$

Dans le cas d'un problème elliptique, on peut avantageusement prendre la norme (isotrope)

$$\mathbf{M}_{\infty}(p)=\frac{1}{\varepsilon}\cdot\mathbf{I}\cdot\max_{i}(|\lambda_i|)$$

Erreur d'interpolation en 2D

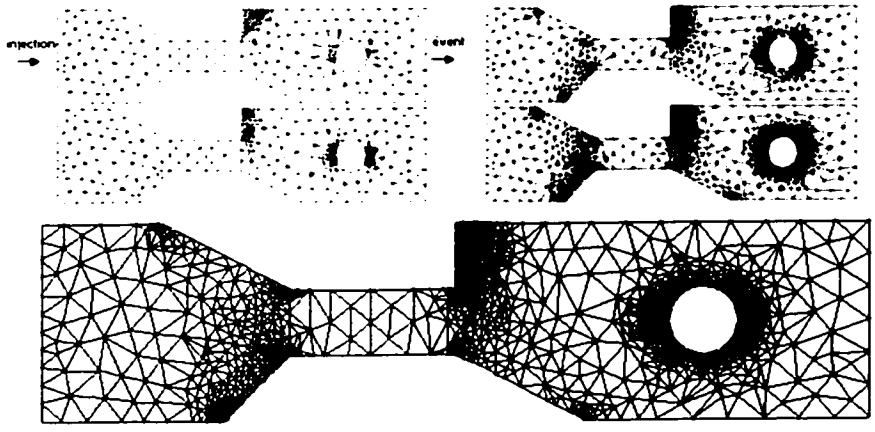
Calcul de la matrice hessienne par « lissage des contraintes »



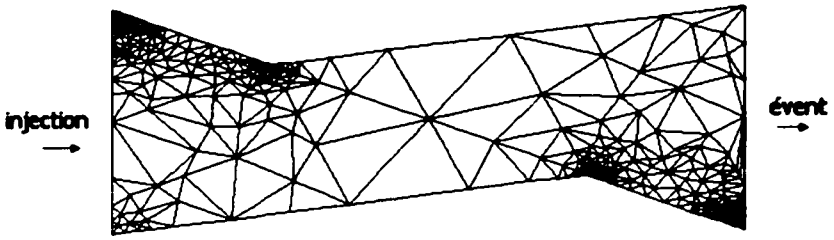
$$H(p) = \begin{bmatrix} \frac{\partial^2 p}{\partial x^2} & \frac{\partial^2 p}{\partial y \partial x} \\ \frac{\partial^2 p}{\partial x \partial y} & \frac{\partial^2 p}{\partial y^2} \end{bmatrix}$$

Erreur d'interpolation en 2D

Convergence

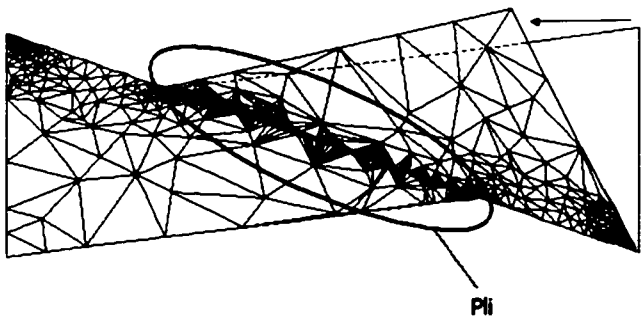


Erreur d'interpolation en 2D



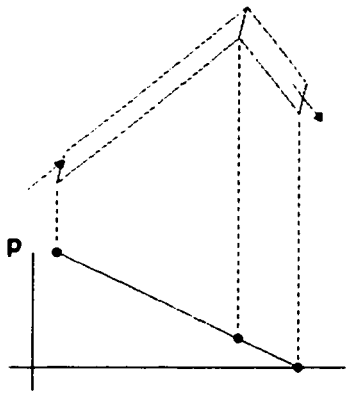
Erreur d'interpolation en 3D (surfaces)

Application de l'estimateur

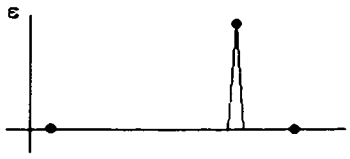


Erreur d'interpolation en 3D (surfaces)

Changer l'évaluation de l'erreur



Le champ de pression est directement représentable à l'aide d'éléments linéaires.
Mais
L'estimateur 2D renvoie une erreur non nulle au voisinage du pli.



Erreur d'interpolation en 3D (surfaces)

Changer l'évaluation de l'erreur

L'estimateur de l'erreur ε vaut :

$\varepsilon_t = \varepsilon_g + \varepsilon_f$ incluant erreur géométrique et erreur fonctionnelle.

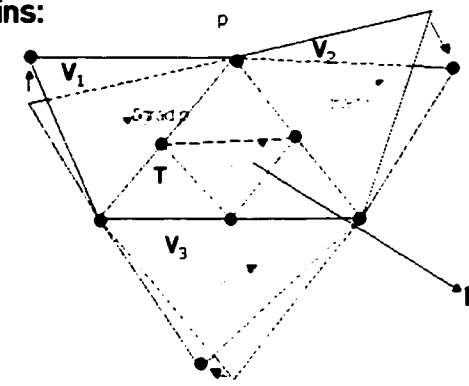
-> l'erreur géométrique peut être soit :

- déterminée par d'autres moyens
- imposée par le maillage de fond (fich. STL p. ex.)

Seule l'erreur fonctionnelle importe ici.

Erreur d'interpolation en 3D (surfaces)

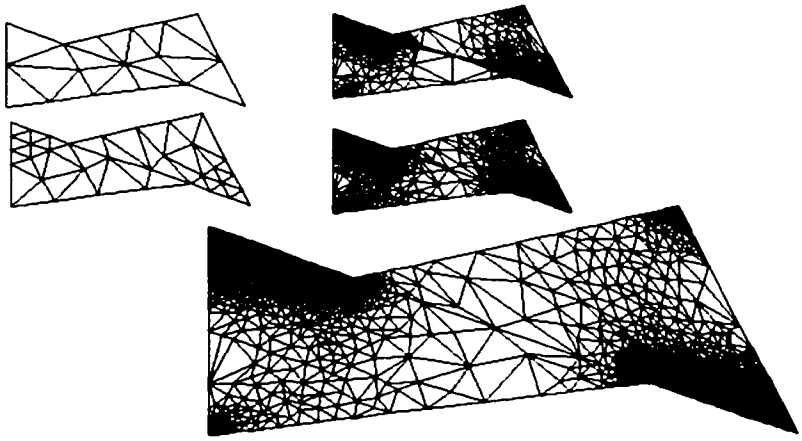
L'erreur fonctionnelle ε_f se calcule en ramenant dans le plan les gradients calculés dans les éléments voisins:



$$\mathbf{H}(p) = \begin{bmatrix} \frac{\partial^2 p}{\partial x^2} & \frac{\partial^2 p}{\partial y \partial x} \\ \frac{\partial^2 p}{\partial x \partial y} & \frac{\partial^2 p}{\partial y^2} \end{bmatrix}$$

Erreur d'interpolation en 3D (surfaces)

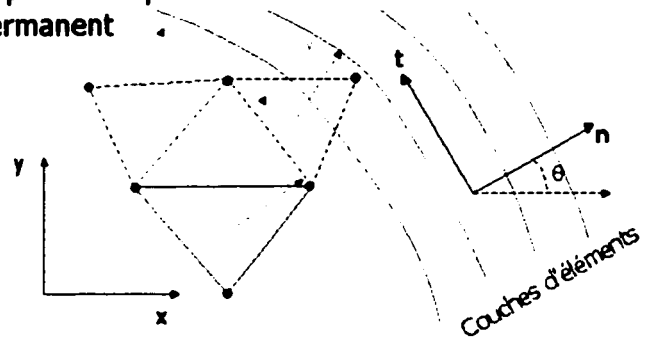
Résulats



Déplacement du front

But : un maillage épousant les positions successives du front

Hypothèse : prédiction par une simulation d'écoulement en régime permanent



Déplacement du front

La métrique gouvernant le maillage est (2D) :

$$\mathbf{M}_n(\mathbf{v}) = {}^t\mathbf{R} \cdot \begin{bmatrix} \frac{1}{\|\mathbf{v}\|^2} & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{\alpha^2} & 0 \\ 0 & \frac{1}{\beta^2} \end{bmatrix} \cdot \mathbf{R} \quad \text{avec} \quad \mathbf{R} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

α est le délai désiré entre deux pas de temps (couche d'éléments)

β est la taille désirée dans le plan normal à la vitesse (tangent au front)

Déplacement du front

β choisi pour obtenir un maillage isotrope

3653 éléments

β choisi pour obtenir un maillage anisotrope

1388 éléments

Déplacement du front

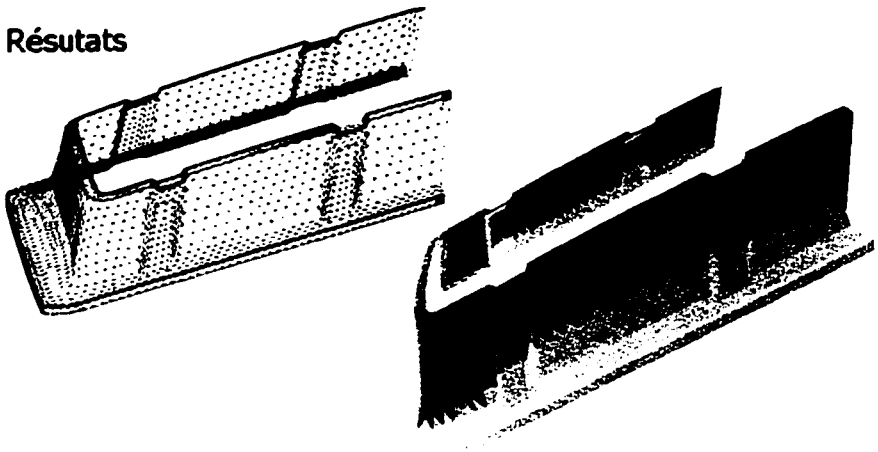
α, β bornés par l'estimateur d'erreur

$$\mathbf{M}_s(p,\mathbf{v}) = {}^t\mathbf{R}\cdot\begin{bmatrix} \max\left(\frac{1}{(\|\mathbf{v}\|\cdot\alpha)^2}\cdot\frac{1}{\varepsilon}\max_i(|\lambda_i|)\right) & 0 \\ 0 & \max\left(\frac{1}{\beta^2}\cdot\frac{1}{\varepsilon}\max_i(|\lambda_i|)\right) \end{bmatrix}\cdot\mathbf{R}$$

3000 éléments

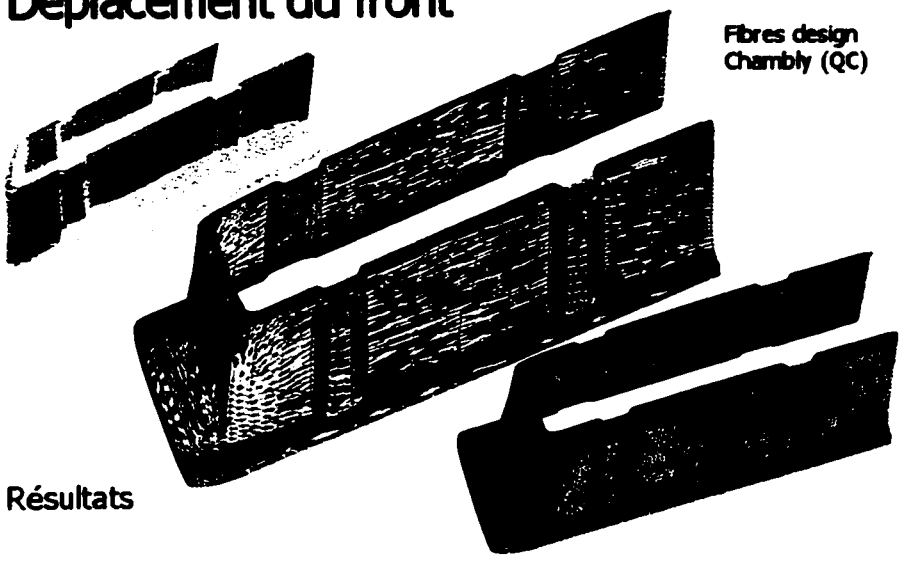
Déplacement du front

Résutats



Déplacement du front

Fibres design
Chambly (QC)



Résultats

Nombre de Courant

Similaire au cas 7.2; connaissant le pas de temps Δt, et le nombre de courant désiré:

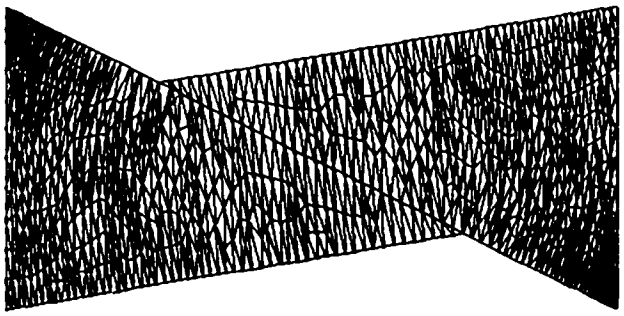
$$C_g = \frac{\|\mathbf{v}\| \cdot \Delta t}{\Delta x}$$

la métrique peut être exprimée :

$$\mathbf{M}_h(\mathbf{v}) = {}^{\mathrm{T}}\mathbf{R} \cdot \begin{bmatrix} \left(\frac{C_g}{\|\mathbf{v}\| \cdot \Delta t}\right)^2 & 0 \\ 0 & \frac{1}{\beta^2} \end{bmatrix} \cdot \mathbf{R}$$

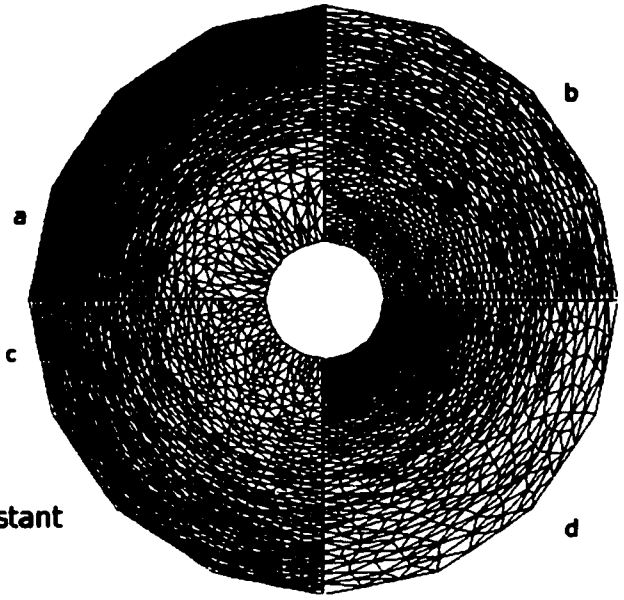
Nombre de Courant

Résultat



Comparaison

Point d'injection
(pression
constante)



a- Δt constant
b- Δx constant
c- nb de Courant constant
d- erreur constante

Comparaison

Incompatibilité entre

- un maillage rendant l'erreur d'interpolation uniforme
- un maillage rendant le nombre de Courant uniforme

8- Conclusion

- Méthode de maillage anisotrope indépendante de la représentation géométrique exacte (fichier STL) – article 1.
- Déplacement du front par une méthode de surfaces de niveaux (level-sets) combiné avec remaillage interactif et anisotrope – article 2.
- Application au procédé RTM et validation expérimentale (cas isotherme) – article 2.

8- Conclusion

- Application au procédé RTM (cas non isotherme) : Problèmes de stabilité – article 3.
- Maillage initial adapté selon un critère d'évolution du front – article 3.
- Maillage initial améliorant la condition CFL pour les problèmes de transport – articles 3 et 4.
- Estimateur d'erreur adapté aux surfaces discrètes – article 4.

9- Recommandations

- Extension tridimensionnelle (mailleur, level-sets)
- Conditions aux limites au niveau du front
- Conservation de la quantité de resine
- Présimulation rapide
- remaillage si déformation du domaine (inserts, moule flexible...)
- Estimateur d'erreur en thermique
- Approche semi-lagrangienne pour le transport.