Rapport Annuel d'avancement de thèse
1ère année (2004/2005)

Génération Automatique de Topologie de maillages

Thèse LRI/Airbus (Université Paris XI)

Thésard : Pierres Matthieu

Directeurs de thèse: Marc Schoenauer
Michèle Sebag
Responsable Airbus : Bernard Marquez

Présentation

La génération de topologie pour un maillage structuré est une tâche actuellement effectuée manuellement par une équipe de ‘mailleurs’. La topologie définie un ensemble de blocs liés entre eux, découpant la partie extérieure d’un modèle CAO d’avion ou de parties d’avion (pour airbus). Cette topologie est utilisée afin de créer un maillage structuré hexa. Chaque bloc de topologie est découpé en cellules, en suivant des lois de distributions suivant les arêtes des blocs. L’ensemble de ces cellules forment le maillage structuré (c'est-à-dire l’intersection entre deux cellule est une et une seule facette). Ce maillage est ensuite utilisé afin d’effectuer des calculs CFD en volumes finis.

La qualité de la topologie influe donc sur le maillage généré, et par suite sur la qualité du calcul voire même la convergence du calcul. Il s’agit donc d’une étape cruciale qui dans le domaine aéronautique est encore manuelle. La génération d’une topologie complète peut durer entre 1 demi-journée pour les plus simples jusqu'à plusieurs semaines pour certaines. Cela constitue donc un goulet d’étranglement d’un point de vue industriel.

Cette génération se fait sous diverses contraintes suivant les résultats de CFD visés, comme certaines formes de topologies autour de certaines pièces (par exemple des blocs suivant le contours d’une voilure forment un groupe de blocs C). Il y a de plus certaines particularités plus délicates comme certaines structures définie comme optimum par expérience (notamment les pivots saumon, ou les blocs pivots dans les culots en O).

Une topologie est donc définie par un ensemble de blocs et les relations entres les blocs (voisinages). De manière générale les blocs sont des hexaèdres, et peuvent être dégénérés (faces ou arêtes manquantes). Les faces peuvent être décomposées en sous faces représentant chacune l’interface entre deux blocs. Lorsqu’une base topologique contient des blocs possédants plus d’un bloc voisin par face, on appelle cet base une topologie non totalement coïncidente. Dans l’espace topologique un bloc est représenté par un hexaèdre possédant des faces, arêtes, et vertices. Comme les faces et les arêtes peuvent être décomposées, les blocs peuvent être déformés (les arêtes étant des lignes brisées).

Le but de la thèse est d’étudier les possibilités de génération automatiques des topologies proches de celles créée par les experts en maillages. Et ce en respectant des contraintes topologique comme le type de blocs autour des formes.

Topologie

Analyse / choix des données

Les données utilisées sont des bases de données DAMAS. Elles comprennent la description, pour chaque base, d'une topologie complète comprenant la liste de blocs, sous-faces, sous-edges, vertices, ainsi que de la discrétisation complète de l'espace (maillage structuré hexa). Certaines bases contiennent aussi les NURBS représentant les surfaces CAO. Nous n'essayons pas de générer la discrétisation des edges (afin de générer le maillage structuré fin), donc seul une partie des bases nous intéresse.

Les topologies enregistrées ne sont pas coïncidentes totalement (il arrive qu'une face soit découpé en plusieurs surfaces, chacune ayant un bloc différent comme voisin…). Cela risque de poser des problèmes d'apprentissages par la suite. Actuellement chaque bloc possède une liste non ordonnée et de taille variable de sous-faces (elles-mêmes possédant une liste de sous-edges, possédant eux aussi une liste des vertices…).
Afin de palier à ce problème, les bases sont préalablement découpées récursivement, tel que si un bloc comporte une face avec plusieurs sous-faces, il est découpé. Appliqué récursivement, cette méthode nous permet d'avoir une base avec des blocs possédant des caractéristiques communes :

- chaque bloc possède 6 faces, contenant 0 ou 1 sous-face
- chaque face possède 4 edges, contenant 0 ou 1 sous-edge
- chaque edge possède 2 vertices.

Afin de ne pas rater des informations essentiels pour l'apprentissage, toutes les données disponibles concernant les blocs, sous-faces, sous-edges et vertices sont collectées :

Les blocs :

1. identifiant du bloc (un numéro)
2. type du bloc : numéro, il y en a 9 max, en fait 6 (à partir de 31). (et qq part, existe les noms correspondant : 31 = HEXA; 34 = Pyramide, etc)
    a. BM a dit : 6
3. Erikson : coeff de cisaillement, Cell (codage un peu crade) : Erikson du bloc comme un gros tout (le bloc est une seule cellule); suit ensuite, les valeurs min, max, moy, var des Erikson des cellules en lesquelles est discrétisé le bloc pour calculer l'Erikson.
4. Volume Exact: Cell (idem, codage en gros); Reel : somme des volumes des cellules.
5. Volume Mor : (idem Volume, mais autre methode de calcul : Morphologique).
6. Volume Moyen: (idem Volume, mais autre methode de calcul).
7. Volume NMSB : (idem Volume, autre methode de calcul).
8. Faces : il y en a exactement 6 par bloc; mais certaines peuvent etre vides; => blocs dégénérés.
    a. Chaque face : 1 numéro : fixe de 1 à 6; détermine son orientation par rapport au repère intrinseque du bloc.
    b. Nb de sub face : 1 si entierement découpé; mais parfois plusieurs (jusqu'à env. 20+) dans les bases minimisées.
    c. Liste des identifiants des subfaces

Les sous-faces :

Attention, les faces sont des êtres virtuels, le fichier ne contient que les sous-faces; une face de bloc contient une ou plusieurs sous-faces - voire 0.

1. Identifiant
2. Numero d'interface caract : 1=limit, 2=adjacent, 3=adj. non coïncident, on oublie dans un premier temps (adjacence, revoir)
3. Type : valeur multivaluée, 4 possibilités (3 importantes 21=quadra, 22=triangle, 23=segment).
4. Boundary condition: num. entier (ou 0) : le code des conditions limites, à peu près la famille de surfaces géométriques sur laquelle est projetée (collée) la sous-face (topologique) ou alors information topologique liée à la géométrie (sillage, culot...).Si 0 : il s'agit d'une sous-face sans conditions limites (internes). Attention il peut exister plusieurs conditions limites!!.
5. Edge : il y en a toujours 4; mais certains peuvent être dégénérés (absents); et leur numérotation, quoique systématique, est arbitraire (en fonction du repère de la sous face).
    a. numéro (entre 1 et 4)
    b. Nb de sous edges portés par l'edge
    c. Identifiant desdits sous-edges.

Les sous-edges

(Comme pour les faces les edges sont virtuels).

1. Identifiant
2. Numéro d'interface caract : (comme pour les sous-faces)
3. Nombre de vertices : dans tous les cas observés ce nombre est de 2
4. Liste des vertices : Id des vertices ordonnées selon le repère de l'edge (rmin / rmax)
5. Boundary condition: (comme pour les sous-faces). De la même façon il peut exister plusieurs boundary condtions.

Les vertices

1. Identifiant
2. Coordonées : dans le repère xyz (en simple ou double précision, suivant l'information contenue dans le .mesh)
3. Projection : si le point est projeté sur une (ou plusieurs) famille(s) de surface, pour chaque projection on dispose des informations suivantes :
    a. Famille : numéro de la famille de surface (cf .surf)
    b. Num Patch : indice du patch de la surface (donnée CAO). Peu ou pas d'utilité pour notre problème car arbitraire
    c. Type de projection : 1=automatique, 2=manuelle. Dans les cas exemples il n'y a que des projections automatiques
    d. Coordonnées : coordonnées du point sur la surface en xyz (permet de vérifier que le vertex est bien 'réellement' sur la surface)
    e. Normale : Vecteur normale à la surface en ce point (normalisé) dans le repère xyz
    f. UV : coordonnées du point sur la surface en UV coordonnées relative au référentiel surface (mêmes remarques que pour le numéro de patch)

De plus, il faut aussi recueillir les informations concernant la topologie elle-même (description de la scène) :

1. avion concerné (suivant la nomenclature Airbus).
2. informations métriques globales
    a. envergure
    b. longueur
    c. hauteur
    d. corde
    e. surface voilure
    f. …
3. liste des pièces rencontrées (par exemple fuselage, voilure …)
    a. Nom de la pièce
    b. Famille de surface liée (suivant le format des conditions limites)
    c. topologies autour des pièces (C autour de la voilure, …)
    d. spécificités (culot, pivot saumon ….)
4. liste des raccords
    a. les deux pièces concernées
    b. le type de raccord (C45 ou C90)

Extraction

Les données sélectionnées sont parsées de DAMAS vers un format XML simple, et ce afin de pouvoir l'utiliser facilement, et de permettre une lisibilité aisée.

Pour chaque base d'information, un fichier XML distinct est créé :

- .blck : liste des blocs
- .face : liste des faces
- .edge : liste des edges
- .vrtx : liste des vertices
- .def : définition de la base
- .mesh : informations sur la topologie (nombre de blocs, sous-faces…)

Les informations de définissions sont entrées à l'aide d'une interface Tcl/Tk. Celle-ci génère aussi des scripts ksh réutilisables afin d'automatiser l'extraction, qui peut parfois prendre plusieurs minutes (voire plus en cas de découpage de la base).
Les fichiers CAO ne sont pas utilisés dans la phase d'apprentissage et non extraits.

Apprentissage

Choix de la représentation

Nous n'avons pas à notre disposition de base topologique définie comme mauvaise. En effet les exemples existant sont uniquement les topologies correctes et ayant été validées.
On se retrouve dans un cadre d'apprentissage positive-only (que des bons exemples).
De plus, nous disposons d'une quantité limité de topologies valides (une 50 maximum), et le coût de génération à la demande peut être chère voire incompatible avec le planning des mailleurs.
L'exemple de base choisi est donc le bloc. En effet, on a entre 300 et 5000 blocs dans une base entièrement découpée. L'objectif sera donc de tenter de caractériser un bloc par rapport aux blocs observés dans les bases validés.
Il est possible de retrouver des informations concernant le bloc, comme par exemple son type : appartient-il à un groupe de blocs C/H ou O ?
Par la suite, les blocs situés au niveau des jonctions semblent présenter des attributs spécifiques et le choix est donc de séparer les blocs en 5 catégories :
C, H, O, Jonction 45, jonction 90.
Pour être utilisé dans la majorité des algorithmes et programmes d'apprentissage existant, chaque bloc est représenté par un vecteur d'attributs (ou features).

Génération de Bases d'apprentissage

Les données XML sont parsées afin d'être réutilisables et enregistrables sous d'autres formats.
Afin de commencer des tests sur les bases de topologies, un ensemble de descripteurs simples est sélectionné.

Liste des attributs :

- base : nom de la base d'où est issu le bloc (première partie d'une clé unique)
- identifiant : numéro du bloc dans la base (deuxième partie de la clé)
- type: {30-39}
- nb_voisin : nombre de voisins
- erikson : critère d'erickson approché
- vol_exa: volume 'exact' approché
- vol_moy: volume 'moyen' approché
- vol_mor: volume 'mor' approché
- vol_nsm: volume 'nsmb' approché
- vois_erik_moy: moyenne des voisins de l'erickson
- vois_erik_var: variance des voisins de l'erickson
- vois_vexa_moy: moyenne des volumes 'exact' des voisins
- vois_vexa_var: variance des volumes 'exact' des voisins
- vois_vmoy_moy: moyenne des volumes 'moyen' des voisins
- vois_vmoy_var: variance des volumes 'moyen' des voisins
- vois_vmor_moy: moyenne des volumes 'mor' des voisins
- vois_vmor_var: variance des volumes 'mor' des voisins
- vois_vnsm_moy: moyenne des volumes 'nsmb' des voisins
- vois_vnsm_var: variance des volumes 'nsmb' des voisins
- BC_fam_face_1: nombre de face ayant une condition limite de la famille 1 (*)
- BC_fam_face_2: …

- BC_fam_face_3:
- BC_fam_face_4:
- BC_fam_face_5:
- BC_fam_face_6:
- BC_fam_face_7:
- BC_fam_face_8:
- BC_fam_face_9:
- BC_fam_face_10:
- BC_fam_face_11:
- BC_fam_face_12:
- BC_fam_face_13:
- BC_fam_face_9999:
- BC_fam_edge_1: nombre d'edges ayant une condition limite de la famille 1
- BC_fam_edge_2: …
- BC_fam_edge_3:
- BC_fam_edge_4:
- BC_fam_edge_5:
- BC_fam_edge_6:
- BC_fam_edge_7:
- BC_fam_edge_8:
- BC_fam_edge_9:
- BC_fam_edge_10:
- BC_fam_edge_11:
- BC_fam_edge_12:
- BC_fam_edge_13:
- BC_fam_edge_9999:

Les familles de conditions limites définissent le type de condition limite, comme par exemple paroi, sillage, … (pour une liste plus détaillée consulter le fichier climite.cfg)

Tests d'apprentissages

Afin de faire un premier jet de tests des algorithmes sont choisis, ainsi que les programmes existant :

- c4.5 (/ c4.5 rules) : classifieur générant des arbres de décisions (systèmes de règles).
- SVMTorch : classifieur utilisant les Support Vector Machines, nous testerons des noyaux linéaires et gaussiens.
- Roger : générateur évolutionnaire de fonctions de l'ensemble de description vers [0,1], en minimisant l'aire sous la courbe ROC (courbe définie par le nombres d'exemple positifs bien classés par rapport aux exemples négatifs mal classés, et ce suivant toutes les coupes en un point possibles de la fonction générée sur [0,1]), développé au LRI.
- JRip : classifieur par un système de règles (développé sur la plateforme Weka)
- J48 : un équivalent c4.5 (développé sur la plateforme Weka)

Après quelques tests avec plusieurs bases prises conjointement, la plateforme Weka manque de stabilité pour certains runs.

Premier essai de classification : C / non C

Dans un premier temps, il a été décidé de tester l'apprentissage en ce concentrant sur les blocs C (les C, C45 et C90) et ceux qui ne le sont pas (les H et les O).
Les résultats sont très bons, et même trop bon (99% de prédictions correctes en moyenne).
Une cause à cette sur-qualité est observée comme étant liée à la corrélation entre blocs C et conditions limites. Cela est du notamment au fait que, les bases possédées actuellement ne contiennent que des topologies avec des C autour de chacune des pièces présentes.

Sans les conditions limites, les résultats restent corrects (88% en moyenne), mais cela peut être dû à un autre biais. En effet, en raison du découpage complet de la base, les blocs proches de la peau possèdent un petit volume et sont plus soumis aux cisaillements. Mais cette caractéristique est plutôt une définition correcte aux vues des descripteurs actuels.

Classification base réelle / fake

Afin de vérifier si il était possible de caractériser un 'bon' bloc, il est possible de tenter de le discriminer par rapport à un bloc 'mauvais'. Le problème réside dans la génération de ce type de blocs, qui ne doivent pas être trop mauvais non plus.
Afin de palier au problème de loi de distribution des attributs, une base de faux blocs (fakes) est générée en permutant les attributs au sein des exemples d'une base valide. Afin d'éviter l'utilisation par l'algorithme de corrélations trop évidentes certains attributs sont permutés par groupes (notamment les 4 calculs de volumes qui sont à priori corrélés).

Les résultats sont corrects après analyse des règles générées (c4.5 rules) on observe que l'apprentis utilise la corrélation entre les types des familles de conditions limites au niveau des faces et des edges pour le même numéro.

Sans les conditions limites les résultats de c4.5 et SVMTorch (avec un noyau Gaussien) sont corrects (80% en généralisation). Et de plus amples analyses sont nécessaires pour mieux expliquer les résultats obtenus.

Discrimination entre les bases

Afin de tester si les blocs d'une base à une autre possèdent des différences très importantes, il a été essayé de discriminer les blocs par rapport à leur base respective.
Les résultats de la discrimination sont plutôt mauvais (erreur >60%, avec 8 bases).
Cela implique qu'avec ces descripteurs les bases ne sont pas vraiment différentiables, ce qui est un point important pour la suite du travail.

Objectifs à moyen et long termes

- Analyses complémentaires des résultats. Et notamment de l'importance de chaque attributs.
- Récupération des informations de classe O dans les culots.
- Tests d'apprentissage multi classe C/H/O/C45/C90.
- Génération de scripts de visualisations des blocs mal classés (sur quickview)

- Génération de procédure d'ajout/test de nouvelles bases
- Ajout de bases d'apprentissage (au sein d'airbus), afin de palier au manque de cas H/O

- Etudes du système de génération /optimisation (à l'aide du logiciel Hexa)
- Génération de fakes à l'aide de scripts hexa

- Créations de topologies de base (création grossière des C et O autour des formes et premiers plans de coupes)
- Essais d'optimisation de topologies de base