Partie II : Modèlisation géométrique

Dominique Bechmann

Chapitre 1

Modèlisation géométrique par contraintes

1.1. Présentation, motivations et exemples

1.1.1. *Les modeleurs déclaratifs*

Les modeleurs géométriques classiques permettent de décrire des formes géométriques très variées, mais ils ne prennent pas en compte les intentions de l'utilisateur, et n'utilisent pas son langage, ses gestes, le savoir-faire de son métier ; ceci ralentit la définition des maquettes numériques. Les modeleurs déclaratifs (ou la modèlisation par formes caractèristiques (*features*) en CFAO) entendent combler ce manque : idéalement, l'utilisateur (typiquement un groupe d'utilisateurs) spécifie plus ou moins précisément et avec le langage et les gestes qui lui sont familiers les contraintes sur l'objet géométrique ou la scène ; selon l'application, ces contraintes sont des contraintes géométriques, esthétiques (une "ligne de caractère" avec un "beau galbé"), mécaniques, médicales, économiques, etc ; le modeleur traduit ces contraintes en une représentation interne, par exemple un système de contraintes géométriques (ce chapitre ne considère que ce seul cas), ou bien un problème d'optimisation avec contraintes pour l'optimisation de formes, ou un problème d'optimisation multicritère ; le modeleur propose ensuite à l'utilisateur une ou des solutions qui satisfont au mieux les contraintes spécifiées ; l'utilisateur prend connaissance de ces solutions et corrige sa spécification ; une maquette numérique satisfaisante est obtenue après un dialogue entre l'utilisateur et le modeleur.

Les contraintes de la modèlisation déclarative permettent de décrire aisément des scènes ou des décors complexes de la vie quotidienne : disposer des livres sur des

---

Chapitre rédigé par Christophe JERMANN et Dominique MICHELUCCI et Pascal SCHRECK.

rayonnages ; placer des meubles (une table, 4 chaises, une armoire) dans une pièce ; placer des fruits dans cette corbeille ; placer 2 voitures dans le garage ; placer des documents sur un bureau. Ces contraintes sont assez lâches, et s'accomodent fort bien d'un peu de hasard dans les solutions et lors de la résolution.

Ce chapitre ne présente qu'une partie de la modèlisation déclarative : la modèlisation par contraintes géométriques. Avec cette dernière, l'utilisateur ne donne plus les coordonnées des éléments géométriques (points, droites, plans, courbes, surfaces) ; il spécifie des contraintes géométriques sur ces éléments : des relations d'incidences, de parallélismes, de perpendicularité et de tangences, des contraintes métriques de distances et d'angles. La modèlisation par contraintes géométriques est commune à tous les modeleurs déclaratifs.

Typiquement, l'utilisateur entre interactivement une esquisse, et spécifie des contraintes ; parfois, le modeleur devine certaines contraintes, comme des relations d'incidence, de parallèlisme ou de perpendicularité en architecture. Un solveur corrige ensuite l'esquisse, pour satisfaire les contraintes. Aujourd'hui, tous les solveurs analysent d'abord le système de contraintes, pour y détecter de possibles erreurs, et pour planifier la résolution quand le système est correct : le système est décomposé en sous systèmes irréductibles, qui sont résolus dans un certain ordre, puis les solutions des sous systèmes sont assemblées. Selon le contexte, l'utilisateur souhaite une seule solution, celle qui est intuitivement la plus proche de l'esquisse, ou bien toutes les solutions réelles, ou encore toutes les solutions dans un certain intervalle.

Un peu arbitrairement, les méthodes d'analyse des systèmes de contraintes géométriques sont regroupées en trois classes, qui sont : les méthodes géométriques, les méthodes combinatoires, les méthodes numériques probabilistes.

Les méthodes géométriques (§ 1.2) automatisent le raisonnement géométrique ; idéalement, elles produisent un plan de construction, qui enchaîne les constructions géométriques simples : calculer la droite passant par deux points connus, le cercle passant par trois points connus, les points d'intersection entre 2 droites connues, entre 2 cercles connus, etc. Elles s'appliquent typiquement en 2D ou 2D et demi, pour des problèmes solubles à la règle et au compas, mais pas seulement [**?**]. Historiquement, les premiers solveurs dans cette catégorie étaient des systèmes experts ou des programmes Prolog, qui utilisaient des règles géométriques.

Les méthodes combinatoires (§ 1.3) ressemblent aux précédentes, mais elles représentent les systèmes de contraintes par des graphes. Elles y détectent les sous graphes qui représentent les sous systèmes solubles indépendemment. Elles ne retiennent des éléments géométriques (points, droites, plans) que leurs degrés de liberté (le nombre de coordonnées qui les définit, grosso modo) et des contraintes que leurs degrés de constriction, autrement dit le nombre d'équations indépendantes correspondant. Cela ne va pas sans perte de signification ; par exemple, le triangle est bien contraint par 3

contraintes (3 longueurs, 2 longueurs et 1 angle, 1 longueur et 2 angles), mais pas par 3 angles. Autre différence avec les méthodes géométriques, les méthodes combinatoires supposent vérifiée la condition de généricité des paramètres (cf infra).

Les méthodes numériques probabilistes (§ 1.4) considèrent des équations, et non des contraintes ou des graphes. Elles s'appuient sur un témoin, ou exemple ; elles effectuent tous les tests sur le témoin, par exemple les tests d'indépendance des équations (et donc des contraintes), ou les tests de rigidité (cf infra). La plupart du temps, les équations sont obtenues en utilisant des coordonnées cartésiennes ; ainsi une contrainte de distance entre 2 points A et B en 2D donne l'équation : $(x_A - x_B)^2 + (y_A - y_B)^2 - D_{AB}^2 = 0$. Remarquablement, bien que des coordonnées figurent dans cette équation, la valeur de l'expression gauche : $(x_A - x_B)^2 + (y_A - y_B)^2 - D_{AB}^2$ est indépendante du repère. En fait, ceci est vrai pour toutes les équations issues des contraintes géométriques. Cette spécificité explique que la résolution des contraintes géométriques ne se réduise pas seulement à la résolution des systèmes d'équations non linéaires.

La résolution des sous systèmes irréductibles, et l'assemblage des sous figures solutions, utilisent soit des formules explicites pour les problèmes les plus simples (par exemple pour calculer la longueur du troisième côté d'un triangle connaissant les longueurs de 2 autres côtes et 1 angle), soit des méthodes d'analyse numérique (§ 1.5). Les systèmes d'équations algébriques peuvent aussi, en théorie, être résolus par diverses méthodes d'élimination du calcul formel : résultants, bases standard (ou bases de Gröbner), triangulation de Wu-Ritt, méthode de Seidenberg, etc ; de fait, ces méthodes sont utilisées dans les prouveurs géométriques du monde universitaire ; mais elles sont inutilisées dans les solveurs industriels, à cause de leur complexité algorithmique élevée.

Le plus souvent, les contraintes géométriques donnent un système d'équations algébriques : $F(U, X) = 0$, et d'inéquations algébriques : $I(U, X) \geq 0$ sur les coordonnées X des éléments géométriques. U désigne ici les paramètres : les variables dont la valeur est connue juste avant la résolution ; les paramètres sont des distances, des cosinus, ou des grandeurs non géométriques (coûts, forces). Certaines méthodes ne fonctionnent que si les paramètres sont génériques. La généricité garantit que perturber un petit peu les paramètres modifie continuement chaque solution X, autrement dit chaque solution X est une fonction implicite des paramètres. Cette hypothèse de généricité interdit les distances et les angles nuls, ainsi que les angles droits. Seules les contraintes métriques, qui spécifient les distances ou les angles (génériques), utilisent des paramètres ; les relations d'incidence, de parallélisme, de perpendicularité n'en utilisent pas.

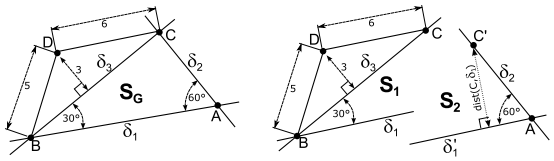

Figure 1.1. *Une esquisse cotée*

1.1.2. *Exemples*

1.1.2.1. *Une épure cotée*

La figure 1.1 peut être construite ainsi : on se donne une droite arbitraire pour δ_3. B est un point quelconque de δ_3. D est un des points d'intersection du cercle centré B de rayon 5 avec une droite parallèle à δ_3 et distante de δ_3 de 3. C est un des points d'intersection de la droite δ_3 et du cercle centré en D de rayon 6. δ_1 est une des 2 droites passant par B, et qui fait un angle de 30 degrés avec δ_3. δ_2 est la droite passant par C et perpendiculaire à δ_1. Finalement, A est le point d'intersection de δ_2 et δ_1.

Plus algébriquement, en posant $u = 5, v = 6, h = 3$, une construction possible est : δ_3 est la droite $y = 0$. $B = (0, 0)$. $D = (\pm\sqrt{u^2 - h^2}, h)$. $C = (x_D \pm \sqrt{v^2 - h^2}, 0)$. La formule pour A existe mais est omise. Cette figure n'est pas la seule solution ; tout déplacement de cette figure est aussi solution.

Pour cet exemple 2D, il existe un plan de construction explicite. C'est le cas chaque fois que le système de contraintes est soluble à la règle et au compas ; un tel système : $F(U, X) = 0$ peut être "triangulé" et la solution s'exprime sous la forme :

$$X_1 = f_1(U), X_2 = f_2(U, X_1), X_3 = f_3(U, X_1, X_2), \dots X_n = f_n(U, X_1 \dots X_{n-1})$$

où les f_i sont des fonctions, éventuellement multivaluées comme la racine carrée ou la fonction $\arccos$, ou bien sous la forme : $P_1 = g_1(U), P_2 = g_2(U, P_1), \dots P_k = g_k(U, P_1, P_2, \dots P_{k-1})$ où les P_i sont des points, définis comme des points d'intersection entre 2 droites, 1 droite et 1 cercle, ou 2 cercles ; les g_i sont, là aussi, des fonctions parfois multivaluées, car 1 droite et 1 cercle (par exemple) se coupent en 2 points. Les fonctions multivaluées posent naturellement une des questions de la modèlisation par contraintes : quelle solution choisir ?

Les méthodes géométriques 1.2, comme la méthode de Sunde, et les méthodes de décomposition (§ 1.3) produisent automatiquement de tels plans de construction, en général moins concis. Les problèmes plus complexes, par exemple en 3D, ne sont plus solubles à la règle et au compas, et il n'existe alors plus de plan de construction aussi explicite. En théorie, les systèmes algébriques $F(U, X) = 0$ restent triangulables, mais cette fois ci sous la forme :

$$f_1(U, X_1) = f_2(U, X_1, X_2) = \dots f_n(U, X_1, X_2 \dots X_n) = 0$$

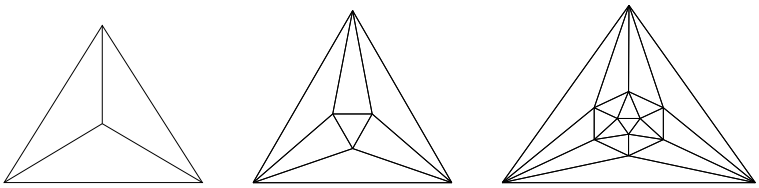

Figure 1.2. *Quelques problèmes de molécules en 3D : le tétraèdre (trivial), l'octaèdre ou plateforme de Stewart, l'icosaèdre.*

La i ème équation ne contient qu'une seule inconnue : X_i. La méthode de Wu-Ritt [CHO 88], ou la méthode de Buchberger, produisent de telles triangulations. Il existe d'autres façons de trianguler, par exemple sous la forme : $E(U,t) = 0, X_i = n_i(U,t)/d_i(U,t), i = 1, \ldots n$, où t est une inconnue auxiliaire, et E, n_i, d_i sont des polynômes. Malheureusement leur complexité, non polynomiale, rend ces méthodes inutilisables.

1.1.2.2. *Les problèmes de molécule*

Dans le problème dit de la molécule, les arêtes d'un graphe non orienté sont étiquetées par des valeurs génériques de distances (Fig. 1.2). Le problème est de trouver un plongement (des coordonnées x_i, y_i, z_i pour les sommets du graphe) qui satisfait les contraintes de distances. Aucune relation d'incidence n'est spécifiée, si ce n'est la dimension k du plongement : typiquement $k = 2$ (en 2D) ou $k = 3$ (en 3D). Ce problème doit son nom à ses applications en chimie moléculaire : reconstituer les configurations d'une molécule à partir de quelques distances entre atomes. Il apparait aussi en robotique, avec la plateforme de Stewart : deux triangles sont connectés par 6 vérins dont la longueur est asservie par un servo-mécanisme ; le tout a la topologie d'un octaèdre. La généricité des paramètres de distance interdit les plongements dégénérés, tels que la colinéarité de 3 sommets ou plus (si $k \geq 2$), la cocyclicité de 4 sommets ou plus (si $k \geq 2$), la coplanarité de 4 sommets ou plus (si $k \geq 3$), etc.

1.1.2.3. *Les problèmes de polyèdre*

Les graphes de la figure 1.3 illustrent d'autres systèmes plus complexes de contraintes géométriques en 3D ; outre les contraintes de distances sur les arêtes, les sommets des faces intérieures, et de la face extérieure, doivent être coplanaires. Les graphes de la figure fournissent des systèmes bien contraints modulo les déplacements, c'est à dire des systèmes n'ayant qu'un nombre fini de solutions, à la position et à l'orientation près en $3D$. Ceci s'étend au graphe de tous les polyèdres eulériens (dont les nombres S, A, F des sommets, des arêtes, des faces vérifient : $S - A + F = 2$).

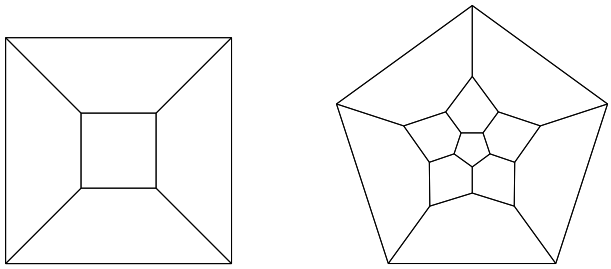

Figure 1.3. *Quelques problèmes polyèdriques en 3D : l'hexaèdre, le dodécaèdre ; ils sont bien contraints.*

1.1.3. *Définitions*

Sauf mention contraire, les contraintes géométriques sont indépendantes du repère : elles ne décrivent que la forme et les dimensions des objets, mais pas leur position et leur orientation dans un repère cartésien. Au mieux, elles ne décrivent donc les objets géométriques qu'à isométrie près (ou modulo les isométries). Les isométries sont les transformations géométriques qui laissent invariantes les angles et les distances ; plus fondamentalement, elles laissent invariantes le produit scalaire ; elles comprennent les anti déplacements et les déplacements. Les déplacements sont les rotations, les translations, et leurs compositions. Un anti déplacement est la composition d'un déplacement et d'une symétrie par rapport à un hyperplan.

Un système bien contraint est un système qui a un nombre fini de figures solutions, modulo les isométries ; on dit aussi : modulo les déplacements, ou à isométrie près, ou aux déplacements près ; souvent, le "modulo les isométries" est sous-entendu. de plus on sous-entend souvent que les contraintes d'un système bien contraint modulo les isométries sont indépendantes (c'est à dire non redondantes). Un triangle contraint par 3 longueurs est un exemple de système bien contraint modulo les isométries ; le fait que les 3 longueurs vérifient ou non les inégalités triangulaires n'est pas pris en compte. Un triangle contraint par 2 angles et 1 longueur, ou bien par 1 angle et 2 longueurs, sont d'autres exemples possibles.

Un système sous contraint est un système qui a un continuum de figures solutions, modulo les isométries ; un triangle contraint par 2 longueurs, ou bien par 2 angles, est un exemple de tels systèmes.

Un système sur-contraint est un système qui n'a pas de solution, ni réelle ni complexe. Un triangle contraint par trois angles est un exemple de système sur-contraint.

Certains systèmes sont bien contraints modulo les similitudes ; les similitudes sont aussi appelées homothéties, ou changements d'échelle. Les similitudes laissent invariants les angles et les proportions (rapports de distances). Un exemple d'un tel système est un triangle contraint par 2 angles.

1.2. Les méthodes géométriques

1.2.1. *La méthode de Sunde*

1.3. Décompositions fondées sur des graphes

La décomposition de systèmes de contraintes géométriques a pour but de ramener leur résolution à celle d'un ensemble de systèmes plus simples : c'est l'application du principe *diviser pour régner*. Par décomposition, on espère améliorer les performances de la résolution et simplifier l'étude des grands systèmes. En effet, les solveurs ont en général une complexité qui n'est pas linéaire dans la taille du système ; par ailleurs, l'analyse de la forme décomposée d'un système permet d'identifier précisément les sous-systèmes qui vérifient ou non certaines propriétés : constriction, redondance, ...

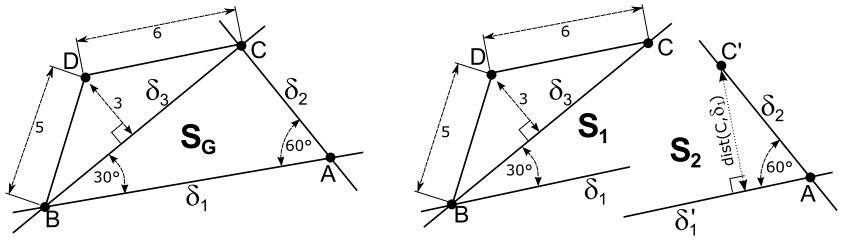

Figure 1.4. à gauche : *une esquisse cotée ;* à droite : *une décomposition*

Considérons l'exemple présenté à la figure 1.4. La partie gauche de cette image représente une esquisse cotée qui spécifie un système de contraintes géométriques (nommé $\mathcal{S}_G$ dans la suite du chapitre) entre 7 entités : les points A, B, C et D et les droites δ_1, δ_2 et δ_3. La partie droite de cette figure présente une décomposition de ce système en 2 parties : les sous-systèmes $\mathcal{S}_1$ et $\mathcal{S}_2$.

Sans nous attarder pour le moment sur la façon d'obtenir cette décomposition, constatons que la résolution des 2 sous-systèmes produit 2 ensembles de sous-figures $\mathcal{F}(\mathcal{S}_1)$ et $\mathcal{F}(\mathcal{S}_2)$ qui peuvent être combinés en faisant coïncider les points C et C' et les droites δ_1 et δ'_1 par déplacement ; ces combinaisons produisent l'ensemble $\mathcal{F}(\mathcal{S}_G)$ des solutions de $\mathcal{S}$. Il est ici garanti que 2 sous-figures pourront toujours être assemblées en une figure solution par l'ajout dans le sous-système $\mathcal{S}_2$ d'une contrainte de distance

entre C' et δ_1' définie comme ayant la même valeur que la distance entre C et δ_1 dans le sous-système $\mathcal{S}_1$. C'est le processus de décomposition qui a automatiquement ajouté cette contrainte, parfois appelée *bord* ou *frontière*, qui vise à représenter $\mathcal{S}_1$ dans $\mathcal{S}_2$. On remarque donc que les sous-systèmes produits par décomposition, s'ils peuvent être résolus séparément, ne sont pas pour autant indépendants les uns des autres ; ici, $\mathcal{S}_2$ ne peut être résolu qu'après $\mathcal{S}_1$ puisqu'il utilise la valeur $dist(C, \delta_1)$ qui devra être lue dans les sous-figures de $\mathcal{F}(\mathcal{S}_1)$. Par ailleurs, l'ajout d'une contrainte dans $\mathcal{S}_2$ fait de lui un système de contraintes géométriques qui n'est pas à proprement parler un sous-système de $\mathcal{S}_G$. Tout ceci nous conduit à une définition générale des décompositions de systèmes de contraintes géométriques :

DÉFINITION.– *Une décomposition d'un système de contraintes géométriques $\mathcal{S}$ est définie par un triplet $(\{\mathcal{S}_1, \cdots, \mathcal{S}_n\}, \prec, \uplus)$ où chaque $\mathcal{S}_i$ est un système de contraintes géométriques appelé composant de $\mathcal{S}$, $\prec$ est un ordre partiel pour la résolution des composants, et $\uplus$ est un opérateur d'assemblage qui étant donné une sous-figure solution $\mathcal{F}_i$ pour chaque composant $\mathcal{S}_i$ produit une figure $\mathcal{F}$ solution de $\mathcal{S}$.*

Cette définition s'applique uniquement aux systèmes bien-contraints (généralement modulo une invariance), chaque composant étant un système lui-même bien-contraint. Il est crucial que chaque composant soit bien-contraint, autrement les ensembles de sous-figures solutions seraient infinis (ou vides) et leur assemblage ne pourrait être calculé. Lorsqu'un système est mal contraint, la décomposition n'a plus de sens : certaines méthodes retournent des composants mal-contraints, d'autres produisent un ensemble de composants bien-contraints mais qui ne couvre pas l'ensemble du système. Cependant, de nombreuses méthodes de décomposition intègrent des mécanismes de détection des cas mal contraints, permettant alors un rattrapage automatique ou un retour à l'utilisateur.

Etudions à présent la façon dont peut être obtenue la décomposition du système $\mathcal{S}_G$. Nous présenterons deux méthodes différentes conduisant à cette décomposition.

1.3.1. *Décomposition récursive descendante*

Le système $\mathcal{S}_G$ a été découpé selon une paire d'entités géométriques (C, δ_1) dupliquée dans les sous-systèmes $\mathcal{S}_1$ et $\mathcal{S}_2$. On peut considérer que cette décomposition a été obtenue en cherchant dans $\mathcal{S}_G$ un *point d'assemblage*, c'est à dire un sous-ensemble d'objets constituant une coupe du système $\mathcal{S}_G$. Ce processus peut être réitéré dans chacun des sous-systèmes résultant d'une coupe, conduisant à la fin à un ensemble de composants indivisibles. Ce mécanisme de décomposition est dit *récursif descendant* en ce sens qu'il part du système initial et le découpe récursivement.

L'élément clé de ces méthodes est l'identification des points de coupe : leur définition doit garantir la possibilité de réassembler les sous-figures une fois les composants

résolus. Usuellement, les points de coupe correspondent aux points d'articulation dans un graphe représentant le système. La figure 1.5–gauche présente le graphe correspondant à $\mathcal{S}_G$.

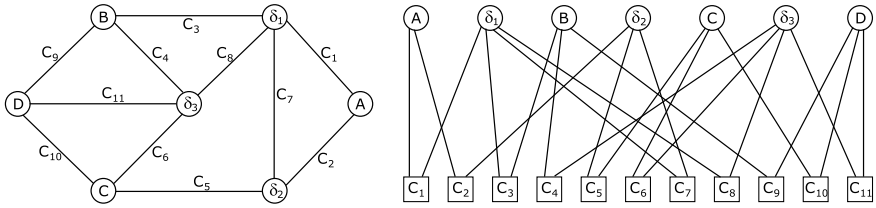

Figure 1.5. *Graphes représentant la structure du système de la figure 1.4*

Dans ce graphe, les sommets sont les entités géométriques et les arêtes représentent les contraintes. Afin d'éviter les hyper-arêtes, on utilise parfois une variante bipartie de ce graphe où objets et contraintes sont des sommets, les arêtes représentant leurs liens (cf. figure 1.5–droite).

Un point d'articulation dans un graphe est un ensemble de sommets dont le retrait coupe le graphe en plusieurs sous-graphes. L'identification d'un tel couple de sommet peut être réalisée en temps polynomial par un algorithme de calcul de connexité. Le couple (C, δ_1) correspond bien à un point d'articulation dans le graphe représentant $\mathcal{S}_G$. Cette paire est dupliquée dans l'ensemble des sous-systèmes produits en découpant $\mathcal{S}_G$: une occurence apparaît dans $\mathcal{S}_1$ et une autre dans $\mathcal{S}_2$. Dans $\mathcal{S}_1$, la position relative de C et δ_1 est bien définie. Du point de vue du graphe, cela s'explique par le fait que le sous-graphe de $\mathcal{S}_1$ est bien biconnecté. En revanche, le sous-graphe restant n'est pas biconnecté, ce qui signifie que la position relative de (C', δ_1') n'est pas bien définie ; autrement dit, ce sous-système est sous-contraint. Afin de maintenir la cohérence entre les deux sous-systèmes, une contrainte $dist(C', \delta_1') = dist(C, \delta_1)$ est introduite dans $\mathcal{S}_2$. De façon générale, une contrainte similaire devra être introduite dans tous les sous-graphes résultant d'une coupe qui ne sont pas biconnectés. Cette contrainte, dénommée *lien virtuel*, établit un lien entre $\mathcal{S}_1$ et $\mathcal{S}_2$ et impose que $\mathcal{S}_1$ soit résolu avant $\mathcal{S}_2$.

La figure 1.6 illustre l'application complète de cette méthode sur le système de la figure 1.4. Les points d'articulation identifiés succéssivement apparaissent en gras, et les liens virtuels en tirets.

La décomposition se termine lorsque tous les sous-systèmes produits sont indivisibles, c'est-à-dire qu'ils correspondent tous à des sous-graphes suffisament connectés (biconnexes dans le cas de $\mathcal{S}_G$). L'ordre entre les sous-systèmes est induit par les liens virtuels. L'assemblage des sous-figures solutions se fait par mise en coïncidence des objets correspondants aux points d'articulation au moyen de déplacements.

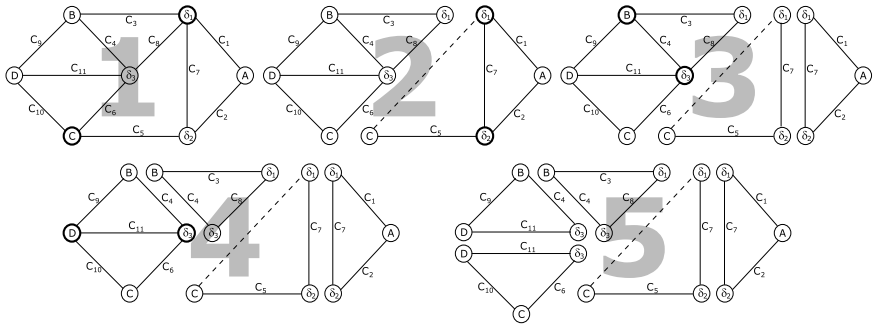

Figure 1.6. *Décomposition descendante du système de la figure 1.4*

La faiblesse de ces méthodes provient de ce que leur caractérisation des points de coupe est incorrecte et incomplète : elles peuvent retourner des points de coupe ne permettant pas l'assemblage (e.g., points confondus, droites parallèles) et ne peuvent parfois plus découper un système alors que cela serait encore possible.

1.3.2. *Décomposition récursive ascendante*

A l'inverse de la décomposition récursive descendante, on pourrait considérer que la décomposition de $\mathcal{S}_G$ a été obtenue en identifiant d'abord le composant $\mathcal{S}_1$ qui constitue un sous-système bien-contraint de $\mathcal{S}$. Une fois ce composant identifié, il a été retiré du système et remplacé par la contrainte de distance entre C' et δ'_1. Le processus a alors été réitéré et a identifié le sous-système $\mathcal{S}_2$ comme un nouveau composant. Ce mécanisme est dit *ascendant* car il identifie d'abord les composants avant de les assembler jusqu'à couvrir l'ensemble du système. L'étape clé des méthodes de ce type consiste à rechercher un petit composant bien-contraint. Cette recherche peut s'effectuer en utilisant des modèles de sous-systèmes bien-contraints répertoriés (méthodes à base de règles, ou de modèles), ou bien à l'aide d'une propriété structurelle (méthodes structurelles) dérivée d'une caractérisation de la rigidité due à Laman.

THÉORÈME.– *(*Laman, 1970*) Un système de n points liés par m distances en 2D est rigide ssi $m = 2n - 3$ et pour tout sous-système de n' points induisant m' distances, $m' \leq 2n' - 3$.*

Ce théorème ne s'applique qu'en 2D et uniquement pour les systèmes de points liés par des distances indépendantes (ou *génériques*), ce qui assure que chaque figure solution est une fonction continue des valeurs de distance et interdit, par exemple, les points alignés ou confondus. Toutes les généralisations de ce théorème proposées à ce jour se sont avérées incorrectes. Malgré tout, une caractérisation approximative appelée *rigidité structurelle* est souvent utilisée. Elle repose sur un compte des *degrés*

de liberté (DDL). Les DDL des objets représentent le nombre de mouvements indépendants qu'ils admettent ; les contraintes retirent des DDL aux objets sur lesquels elles portent. Le nombre de DDL d'un système est la somme des DDL de ses objets privée de la somme des DDL retirés par ses contraintes. Il représente le nombre de mouvements indépendants qu'il admet.

L'intuition de la rigidité structurelle est qu'un système n'est rigide que s'il admet comme mouvement tous, et uniquement, les déplacements. En 2D et 3D, il y a respectivement 3 et 6 déplacements indépendants. Un système est donc structurellement rigide s'il admet 3 DDL en 2D (resp. 6 DDL en 3D) et que ses sous-systèmes en admettent au moins autant. Cette caractérisation est incorrecte car elle omet deux possibilités : la présence de contraintes redondantes (qui retirent conjointement moins de DDL que leur somme) et de contraintes singulières (qui retirent plus de DDL que leur somme). De nombreuses heuristiques ont été proposées pour traiter ces cas particuliers, sans toutefois parvenir à faire de la rigidité structurelle une caractérisation exacte de la rigidité.

Une fois un sous-système rigide identifié, les méthodes ascendantes le remplacent dans le système à décomposer par un *représentant*, qui est le plus souvent un sous-ensemble de ses objets liés par des contraintes rigidifiantes ; on parle de *frontière*.

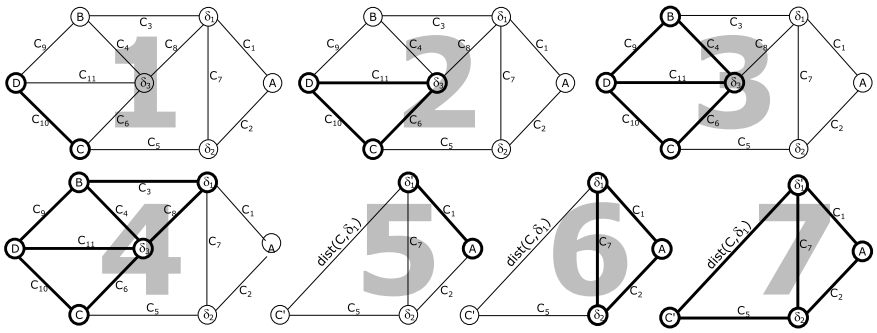

Figure 1.7. *Décomposition ascendante du système de la figure 1.4*

La figure 1.7 illustre la décomposition récursive ascendante de $\mathcal{S}_G$ en utilisant le graphe introduit à la figure 1.5. A chaque étape, le sous-graphe correspondant au composant rigide identifié apparaît en gras. Initialement, c'est le sous-graphe induit par $\{C, D\}$ qui est identifié ; effectivement, un point en 2D offre 2 DDL et une contrainte de distance en retire 1 : on obtient bien un sous-système ayant $2 + 2 - 1 = 3$ DLL en 2D. La décomposition présentée ici utilise alors une étape d'extension visant à agréger un maximum d'objets, un à un, relativement au composant rigide identifié. Un objet

peut être agrégé ssi son ajout au composant préserve la rigidité structurelle. Chaque extension produite ainsi correspond également à un composant bien-contraint. La droite δ_3 est agrégée au composant $\{C, D\}$: elle sera positionnée en utilisant son incidence au point C et sa distance au point D. Il est alors possible d'agréger le point B, puis la droite δ_1. L'extension est terminée et le sous-graphe induit par $\{B, C, D, \delta_1, \delta_3\}$ est retiré du graphe ; pour la suite de la décomposition, il sera représenté par sa frontière, c'est à dire les objets C' et δ_1' liés par une distance. Cette dernière a pour but de maintenir dans le système restant la position de ces objets qui sera déterminée dans le premier sous-système. La décomposition se poursuit alors de la même façon dans le système restant.

Les méthodes ascendantes identifient donc à chaque étape un composant de la décomposition. L'ordre entre les composants provient des représentants de ceux-ci : un composant ne peut être résolu qu'après tous les composants dont il contient les représentants. Dans notre exemple, $\mathcal{S}_2$ utilise la frontière représentant $\mathcal{S}_1$; $\mathcal{S}_1$ doit donc être résolu avant $\mathcal{S}_2$ afin que cette frontière soit déterminée. La combinaison des sous-figures se fera là encore en faisant correspondre par déplacement celles-ci avec leurs représentants dans d'autres sous-systèmes.

Leur faille provient le plus souvent de la caractérisation approximative de la rigidité qu'elles utilisent : incomplète en cas d'utilisation de modèles de systèmes rigides, incorrectes en cas d'utilisation de propriétés structurelles. Toutefois, elles peuvent s'avérer complètes et correctes pour certaines classes de problèmes, comme les assemblages mécaniques par exemple. Par ailleurs, une caractérisation numérique probabiliste de la rigidité proposée récemment semble prometteuse même si elle n'a pas encore été appliquée dans ce type de méthodes.

1.3.3. *Autres méthodes de décomposition*

D'autres méthodes de décomposition travaillent au niveau du système d'équations représentant le système géométrique. Ces méthodes découpent le système sans dupliquer d'objets géométriques et ne pourraient donc pas produire la décomposition du système $\mathcal{S}_G$ présentée à la figure 1.4.

Certaines identifient simultanément tous les composants en utilisant un algorithme de type couplage maximum sur le graphe représentant le système. Elles associent les composants de la décomposition aux composantes fortement connexes de ce graphe qui est doté d'une orientation induite par le couplage maximum. Ces méthodes ne sont pas récursives : elles produisent tous les composants simultanément. Ceci les rend souvent plus faible que les méthodes récursives. Toutefois, le fait qu'elles travaillent au niveau des équations et des variables peut leur permettre de résoudre des demi-objets par des demi-contraintes, ce que ne pourraient pas faire des décompositions travaillant au niveau géométrique. Enfin, le fait qu'elles identifient les composants à l'aide d'une

propriété de graphe les expose aux mêmes lacunes que les méthodes utilisant une caractérisation structurelle de la rigidité : elles peuvent s'avérer incorrectes en présence de redondances ou de singularités.

D'autres identifient itérativement des composants *libres* ; on dit qu'elles procèdent par *propagation des degrés de liberté*. Intuitivement, un composant est libre si les variables qu'il calcule n'interviennent dans aucune autre contrainte du système. La résolution d'un tel composant n'aura pas d'incidence sur le reste du système. Il peut être retiré pour la suite de la décomposition, entraînant la libération d'autres composants. L'identification des composants libres se fait soit au moyen de modèles de systèmes bien-contraints, soit au moyen d'une propriété structurelle. A l'instar des méthodes récursives ascendantes, la faiblesse de ces méthodes tient principalement à leur caractérisation incomplète (modèles) ou incorrecte (propriété structurelle) de la constriction.

1.3.4. *Pistes de recherche*

Chaque méthode de décomposition a ses avantages. Une piste prometteuse consiste donc à les hybrider. Ainsi, la combinaison d'une approche récursive ascendante structurelle avec une approche à base de couplage maximum permet d'obtenir des composants plus petits ; la combinaison d'approches à base de modèles et de méthodes structurelles permet un gain en correction tout en préservant la complétude.

La faiblesse de nombreuses méthodes provenant de leur caractérisation approximative de la constriction, identifier de nouvelles caractérisations plus fiables est également une piste d'actualité. L'utilisation de la méthode numérique probabiliste semble prometteuse, même si plusieurs questions quand à son application pratique restent en suspens ; il faut aussi étudier son intégration dans les schémas de décomposition existant. Une alternative consiste à détecter, et réparer si possible, les cas posant problème pour les caractérisations actuelles, en particulier la présence de redondances ou de singularités.

L'entrelacement de la résolution et de la décomposition, consistant à résoudre immédiatement un composant dès son identification et à remplacer les variables qu'il calcule par leurs valeurs solutions pour la suite de la décomposition, apparaît comme la seule possibilité pour traiter certains systèmes fortement dégénérés et mérite une étude approfondie. Enfin, la définition de méthodes permettant d'obtenir la solution attendue par l'utilisateur parmi l'ensemble des solutions d'un système géométrique, si elle a déjà été étudié, reste encore ouverte.

1.3.5. *Références bibliographiques*

Les méthodes de décomposition ont fait l'objet d'un article de fond auquel ce chapitre emprunte plusieurs exemples et explications [JER 06]. Cet article regroupe une bibliographie plus importante.

La première méthode descendante, dédiée aux systèmes de points et droites en 2D, est due à Owen [OWE 91]. Ce type de méthodes a été étudié par la suite par Fudos [FUD 97] puis Joan-Arinyo [JOA 03, JOA 04] qui ont défini leurs limites et leurs liens avec les approches ascendantes. Une extension en 3D, toujours basée sur l'analyse de connexité, est due à Gao [GAO 03]. Michelucci a proposé une méthode descendante où l'analyse de connexité est remplacée par un test numérique probabiliste [FOU 05] qui ouvre de nouvelles perspectives.

Les premières approches récursives ascendantes sont basées sur des modèles, représentés sous forme de règles dans des moteurs d'inférence logique [BUT 79, SUN 86, ALD 88, VER 92, DUF 94]. Les premières approches structurelles, quant à elle, reposaient initialement sur des règles structurelles très simples [HOF 95, BOU 95] avant que ne soit introduit un algorithme de flot [HOF 98] permettant de calculer la rigidité structurelle. Depuis, cette caractérisation a été utilisée dans plusieurs algorithmes de décomposition ascendante [HOF 01, JER 02, SIT 03].

Les méthodes à base de couplage maximum sont issues de 2 communautés disctinctes, celle de topologie structurale [HEN 92, LAM 98] et celle de propagation locale [LAT 96, TRO 97, BLI 98]. Quant aux méthodes à base de propagation de degrés de liberté, elles proviennent du travail original de Sutherland [SUT 63], et ont depuis été appliquées avec succès dans différents domaines de l'informatique graphique [HSU 97, TRO 98, SOS 02, TRO 06].

1.4. La méthode probabiliste du témoin

1.4.1. *Limitations des méthodes combinatoires et géométriques*

Aucune méthode combinatoire ou géométrique ne peut détecter toutes les dépendances entre contraintes géométriques. En effet, tout théorème géométrique permet de créer des systèmes de contraintes redondants ou contradictoires. Il suffit de prendre comme contraintes les hypothèses et la conclusion (ou sa négation) du théorème. Citons 3 théorèmes. En 2D et en 3D, d'après Desargues, si deux triangles $p_1p_2p_3$ et $q_1q_2q_3$ sont perspectives (c'est-à-dire que les trois droites p_iq_i sont concourantes), alors les trois points d'intersection $r_{ij} = p_ip_j \cup q_iq_j$ sont alignés. En 2D, d'après Pappus, si les points p_1, p_2, p_3 sont alignés, ainsi que les points q_1, q_2, q_3, alors les trois points d'intersection r_{12}, r_{23}, r_{13} tels que $r_{ij} = p_iq_j \cap p_jq_i$ sont également alignés. Enfin, en 2D, selon le théorème dual de Pappus, deux triangles perspectives

de deux façons le sont de trois. Ces théorèmes n'utilisent que des relations (ici d'incidences), et donc aucun paramètre de distances ou d'angles (contrairement au théorème de Pythagore). En conséquence, ils s'appliquent à tout système de contraintes géométriques contenant des relations (incidence, parallélisme, orthogonalité), qui sont indispensables en CFAO, et aucune condition de généricité des paramètres ne permet de se prémunir contre eux.

1.4.2. *Principe du témoin*

Cependant, si un témoin, ou exemple, est disponible, il est possible de détecter ces dépendances. Soit $F(U, X) = 0$ le système des équations à résoudre, U le vecteur des paramètres : distances, angles ou valeurs non géométriques (poids, forces, coûts). Un témoin est un couple (V, X_V) tel que tel que $F(V, X_V) = 0$, où $V \neq U$. En d'autres termes, un témoin est une solution d'une variante du système à résoudre. Le témoin et la cible (la configuration inconnue que l'on cherche) ont les mêmes propriétés. Typiquement, l'esquisse saisie interactivement par l'utilisateur est souvent un témoin ; les relations sont vérifiées, les points qui doivent être alignés ou coplanaires dans la solution le sont déjà, et seuls les angles et distances ont besoin d'être corrigés par le solveur. D'ailleurs, pour tous les exemples de ce chapitre, il est facile de trouver un témoin, avec en sus des coordonnées rationnelles. Par exemple, pour les problèmes de molécule, il suffit de tirer des points au hasard et d'en déduire les distances ; pour le problème du dodécaèdre ou de l'hexaèdre, il suffit de tirer les plans des faces au hasard et d'en déduire les sommets et les distances.

Soit (V, X_V) un témoin. L'idée essentielle est de calculer les vecteurs $\dot{X}$ des mouvements infinitésimaux libres $\epsilon \times \dot{X}$ du témoin, tel que le témoin perturbé $X_V + \epsilon\dot{X}$ satisfasse toujours les contraintes spécifiées : $F(V, X_V + \epsilon\dot{X}) = 0$. Avec un développement de Taylor, $F(V, X_V + \epsilon\dot{X}) = F(V, X_V) + \epsilon F'(V, X_V)\dot{X}^t + O(\epsilon^2)$. Ainsi $F'(V, X_V)\dot{X}^t$ doit s'annuler : les mouvements libres sont donnés par le noyau de la jacobienne $F'(V, X_V)$ du témoin. Ils sont classés en deux catégories : les déplacements infinitésimaux qui ne déforment pas la figure, et les flexions qui déforment la figure. Il est possible de donner une base *a priori* des déplacements infinitésimaux libres en 2D. Elle contient t_x une translation sur l'axe des x, t_y une translation sur l'axe des y et r_{xy} une rotation autour de l'origine. La voici :

	$\dot{x_i}$	$\dot{y_i}$	$\dot{a_l}$	$\dot{b_l}$	$\dot{c_l}$	$\dot{u_k}$	$\dot{v_k}$
t_y	1	0	0	0	$-a_l$	0	0
t_x	0	1	0	0	$-b_l$	0	0
r_{xy}	$-y_i$	x_i	$-b_l$	a_l	0	$-v_k$	u_k

(x_i, y_i) représentent un point, (a_l, b_l, c_l) représentent la droite d'équation : $a_l x + b_l y + c_l = 0$, et (u_k, v_k) est un vecteur. Les variables pointées représentent la valeur du

déplacement infinitésimal associé. Les autres inconnues géométriques (coordonnées barycentriques, produits scalaires, aires), et non géométriques (coûts, forces) sont invariantes par déplacement infinitésimal, donc la valeur correspondante dans tous les vecteurs de la base est nulle. On définit de même une base de 6 déplacements infinitésimaux en 3D, constituées de 3 translations en x, y, z et de 3 rotations dans les plans Oxy, Oxz, Oyz.

Une figure ou une sous figure est décrite par un sous ensemble Y des variables (chaque variable correspond à une colonne dans la table précédente). Son degré de déplacements est le nombre de déplacements infinitésimaux indépendants dans le sous tableau $D[Y]$; par exemple, pour une droite en 2D décrite par $Y = (a_L, b_L, c_L)$, le sous tableau $D[Y]$ est :

	$\dot{a}$	$\dot{b}$	$\dot{c}$
t_x	0	0	$-a_L$
t_y	0	0	$-b_L$
r_{xy}	$-b_L$	a_L	0

et son rang est 2 : on détecte bien que ce sont les deux translations t_x et t_y qui sont dépendantes. L'interrogation d'un témoin donne le degré de déplacements de n'importe quelle figure, et aucune condition de généricité n'est requise. De même, le témoin permet d'effectuer les tests suivants :

- Une équation géométriquement correcte doit être indépendante du repère. L'équation : $(x_A - x_B)^2 + (x_A - x_B)^2 - D_{AB}^2 = 0$ est correcte ; l'équation : $x_A = 0$ ne l'est pas. Une équation est indépendante du repère ssi si son vecteur gradient, au témoin, est orthogonal à tous les vecteurs de la base des déplacements.

- Les équations sont indépendantes entre elles (ni redondantes, ni contradictoires) ssi leurs vecteurs gradients au témoin sont indépendants.

- Une figure, ou une sous figure décrite par un sous ensemble Y des variables, est bien contrainte m.i., ou rigide, ssi l'espace vectoriel de ses mouvements infinitésimaux $M[Y]$ est égal à l'espace vectoriel $D[Y]$ de ses déplacements infinitésimaux, et que le degré de déplacements de Y est 3 en 2D, 6 en 3D. On illustre la méthode sur un exemple 2D.

1.4.3. *Un exemple de systéme 2D dépendant, et une preuve probabiliste*

La distance OA est spécifiée par un paramètre u. Le point O est le milieu des points A et B. La distance OC est égale à la distance OA. AC et BC sont orthogonaux ; cette dernière contrainte résulte des précédentes. Un témoin possible est : $O = (0, 0), A = (-10, 0), B = (10, 0), C = (6, 8), u = 10$. Le système d'équations est :

$$\begin{cases} (x_A - x_O)^2 + (y_A - y_O)^2 - u^2 = 0 \\ 2x_O - x_A - x_B = 0 \\ 2y_O - y_A - y_B = 0 \\ (x_C - x_O)^2 + (y_C - y_O)^2 - (x_A - x_O)^2 - (y_A - y_O)^2 = 0 \\ (x_C - x_A)(x_C - x_B) + (y_C - y_A)(y_C - y_B) = 0 \end{cases}$$

Le jacobien est omis par concision. Voici une base possible pour les mouvements infinitésimaux (le noyau du jacobien) :

	$\dot{x_O}$	$\dot{y_O}$	$\dot{x_A}$	$\dot{y_A}$	$\dot{x_B}$	$\dot{y_B}$	$\dot{x_C}$	$\dot{y_C}$
t_x	1	0	1	0	1	0	1	0
t_y	0	1	0	1	0	1	0	1
r_{xy}	$-y_O$	x_O	$-y_A$	x_A	$-y_B$	x_B	$-y_C$	x_C
fl	0	0	0	0	0	0	$y_O - y_C$	$x_C - x_O$

Cette base a pour rang 4 : 3 déplacements plus 1 flexion : C peut tourner autour de O. Donc la figure est flexible. Consulter le témoin montre aussi que $\{O, A, B\}$ est rigide, ainsi que $\{O, C\}$, qu'une des équations est redondante, et même que la dernière équation résulte des autres, en procédant comme suit.

Quand on conjecture que $C(X) = 0$ résulte de $F(X) = 0$, on vérifie d'abord que la conjecture est bien vérifiée dans le témoin, *ie* que $C(X_V) = 0$; sinon la conjecture est clairement fausse ; puis, pour ne pas être abusé par les faux témoins, on vérifie que $C(X_V + \epsilon\dot{X}) = 0$ est toujours vrai pour tous les vecteurs $\dot{X}$ dans la base des mouvements libres du témoin X_V. En utilisant Taylor, $C(X_V + \epsilon\dot{X}) = C(X_V) + \epsilon C'(X_V)\dot{X}^t + O(\epsilon^2)$; ainsi le vecteur gradient $C'(X_V)$ doit être orthogonal à $\dot{X}$, autrement dit, $C'(X_V)$ doit être dans l'espace vectoriel engendré par le jacobien $F'(X_V)$. Ce dernier test détecte les faux témoins ; par exemple, si OC et OA sont perpendiculaires dans le témoin de l'exemple précédent, cette procédure détecte que cette perpendicularité est accidentelle et ne résulte pas du système, car le vecteur gradient de l'équation $\vec{OC}.\vec{OA} = 0$ n'est pas orthogonal au vecteur de la flexion.

D'autres exemples typiques et simples de configurations 3D, où la méthode du témoin détecte les dépendances contrairement aux méthodes basées sur les graphes, sont donnés dans la Fig. 1.8. La configuration la plus à gauche est classique, et connue sous le nom de la double banane [J.G 93] ; les 3 autres figures sont dues à Ortuzar ; 4 sommets n'y sont jamais coplanaires.

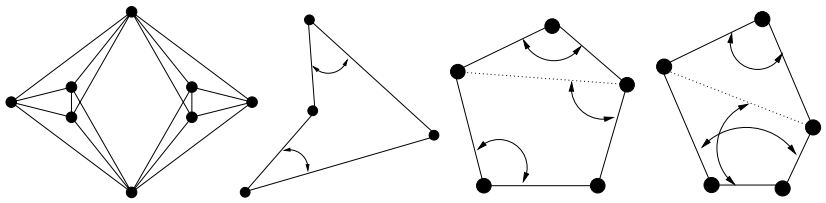

Figure 1.8. *En 3D, la double banane, et trois configurations d'Ortuzar.*

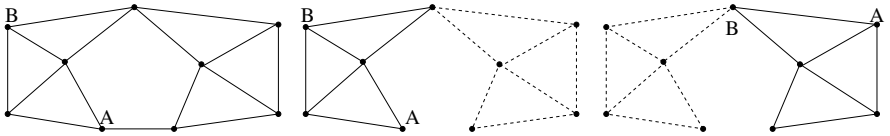

Figure 1.9. *Un système de contraintes rigide 2D. Retirer une contrainte créé un système flexible avec deux parties rigides maximales. Dans les 3 figures, la partie rigide maximale contenant A et B est en traits pleins.*

1.4.4. *Calcul des rangs*

Les seul calculs nécessaires sont ceux du rang (ou du corang) d'un ensemble de vecteurs dont toutes les coordonnées sont des nombres connus. Ces rangs sont calculés par une triangulation de Gauss, ou une décomposition LU. L'imprécision numérique rend délicat le calcul du rang ; par exemple, en arithmétique flottante, le vecteur $(1, 1)$ et le vecteur normé $(\sqrt{2}/2, \sqrt{2}/2)$ ne sont pas exactement colinéaires. Mais comme le témoin est à coordonnées rationnelles, la triangulation de Gauss ou la décomposition LU sont calculables exactement et rapidement ; par exemple les calculs sont effectués modulo un nombre premier proche d'1 milliard de façon probabiliste [MAR 71, SCH 80].

1.4.5. *Décomposition avec le témoin*

Une ancre est un sous ensemble Y de 3 variables en 2D, 6 variables en 3D, qui est rigide (cf supra). Par exemple, en 2D, si le système fixe directement ou indirectement la distance entre les points A et B, alors $\{x_A, y_A, x_B\}$ est une ancre. Clairement un système contient un nombre polynomial d'ancres. Toute ancre Y est contenue dans une figure rigide qui est maximale pour l'inclusion : pour la trouver, il suffit d'adjoindre à Y toutes les variables $x \notin Y$ telles que $Y \cup \{x\}$ reste rigide. Une méthode de décomposition, illustrée Fig. 1.9, s'ensuit : si le système est flexible, déterminer ses parties rigides maximales, en partant de toutes les ancres possibles Si le système est rigide, enlever chaque contrainte à son tour pour rendre le système flexible, et trouver ses parties rigides maximales. Les optimisations sont omises par concision.

1.5. Méthodes de résolution numérique

Les solveurs des modeleurs géométriques utilisent des formules pour résoudre les problèmes les plus simples (tel que : trouver la longueur du troisième coté d'un triangle, connaissant les longueurs des deux autres côtés et un angle) et des méthodes numériques pour les autres.

1.5.1. *Méthode de Newton Raphson*

La méthode de Newton-Raphson résout les systèmes d'équations, linéaires ou non linéaires, algébriques ou transcendantes, à partir d'une approximation d'une solution. Une telle approximation est disponible dans l'utilisation typique d'un modeleur géométrique par contraintes : l'utilisateur saisit interactivement une esquisse (*a sketch*) sur laquelle il spécifie des contraintes ; les coordonnées dans cette esquisse donnent une approximation initiale de la solution désirée ; les contraintes donnent les équations ; ces contraintes sont indépendantes du repère et ne fixent donc les coordonnées qu'à un déplacement près dans le plan ou dans l'espace ; nous supposons ici que 3 équations en 2D (par exemple : $x_1 = y_1 = y_2 = 0$) et 6 équations en 3D (telles $x_1 = y_1 = z_1 = y_2 = z_2 = z_3 = 0$) ont été ajoutées, pour que le système d'équations ait autant d'équations que d'inconnues.

Si X_0 est une approximation d'une racine de $F(X) = 0$, alors $X_1 = X_0 + \Delta_X$ est meilleure ssi $F(X_0 + \Delta_X) \approx F(X_0) + \Delta_X F'(X_0) = 0$. Les inconnues Δ_X sont solutions d'un système linéaire. La matrice $J = F'(X_0)$ des dérivées est appelée la jacobienne : J_{lc} est la dérivée de la c ième équation par rapport à la l ième variable. Son déterminant est le jacobien. Si le jacobien est non nul, la jacobienne est inversible et $X_1 = X_0 - F(X_0)(F'(X_0))^{-1}$. A partir de X_0, l'itération de Newton-Raphson itère donc : $X_{n+1} \leftarrow N(X_n) = X_n - F(X_n)(F'(X_n))^{-1}$ jusqu'à convergence. Il n'est pas indispensable de calculer l'inverse de la jacobienne ; il suffit de résoudre le système linéaire, par la méthode de Gauss, une décomposition LUP, une décomposition en valeurs singulières, etc.

Par exemple, pour un système avec 2 équations f et g, et 2 inconnues x et y : $f(x', y') = f(x + \Delta_x, y + \Delta_y) \approx f(x, y) + \Delta_x f'_x(x, y) + \Delta_y f'_y(x, y) = 0$ et $g(x', y') = g(x + \Delta_x, y + \Delta_y) \approx g(x, y) + \Delta_x g'_x(x, y) + \Delta_y g'_y(x, y) = 0$. Donc

$$(f(x_0, y_0), g(x_0, y_0)) + (\Delta_x, \Delta_y) \begin{pmatrix} f'_x(x_0, y_0) & g'_x(x_0, y_0) \\ f'_y(x_0, y_0) & g'_y(x_0, y_0) \end{pmatrix} = (0, 0)$$

La méthode de la sécante est une variante de la méthode de Newton-Raphson où la jacobienne n'est pas remise à jour à chaque itération.

Quand le point initial est proche, en un certain sens, d'une racine, la convergence de la méthode de Newton-Raphson est quadratique : chaque itération "double

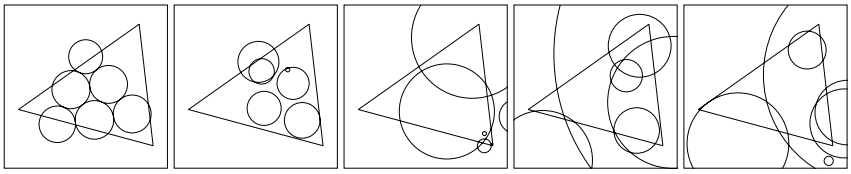

Figure 1.10. *Echec de la méthode de Newton. Voir Fig. 1.11.*

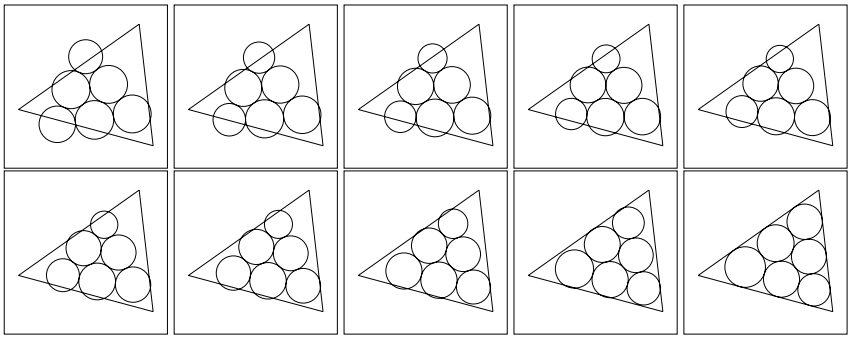

Figure 1.11. *Succès de l'homotopie. Toutes les tangences ont été spécifiées.*

le nombre correct de décimales". Les bassins d'attraction de cette méthode sont des fractales (Fig. 1.12).

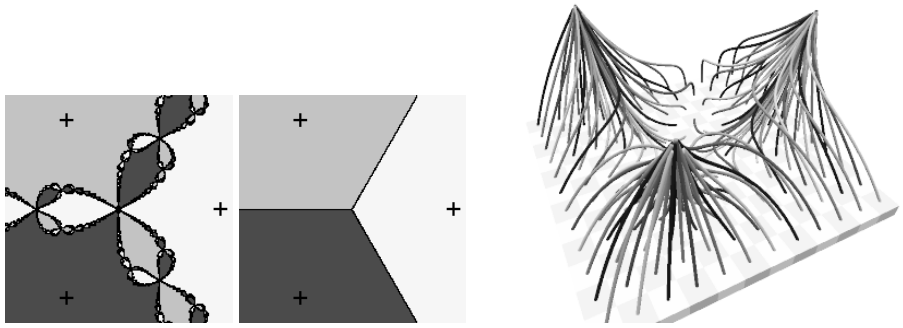

Figure 1.12. *A gauche : bassins (fractales) d'attraction de la méthode de Newton-Raphson ; au milieu, bassins (semi-algébriques) de l'homotopie ; à droite, un échantillon des courbes d'homotopie ; le système est :* $z^3 - 1 = 0, z \in \mathbb{C}$.

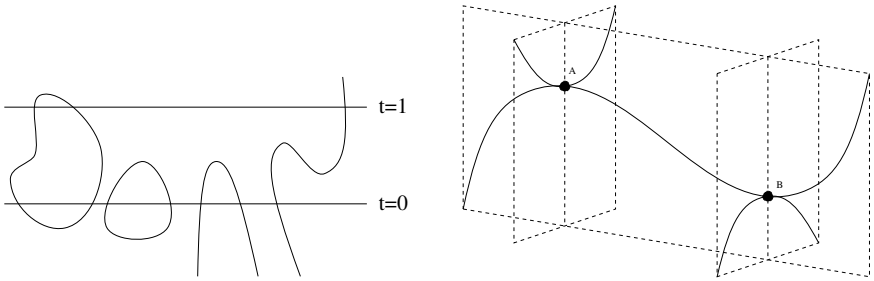

Figure 1.13. *A gauche, quelques topologies possibles pour les courbes homotopiques. A droite, une courbe d'homotopie réelle, en trait gras ; A et B sont deux coudes (turning points). En trait fin les branches complexes. Equations :* $F(X) = 3X^3 - X + \frac{1}{2}$, $G(X) = 3X^3 - X - \frac{1}{2}$, $H(t, X) = 3X^3 - X + \frac{1}{2} - t$, $A_t = \frac{13}{18}$, $A_X = -\frac{1}{3}$, $B_t = \frac{5}{18}$, $B_X = \frac{1}{3}$.

1.5.2. *Homotopie*

La méthode de Newton-Raphson ne converge pas toujours vers la racine attendue par l'utilisateur. L'homotopie est une variante de la méthode de Newton-Raphson, au comportement plus intuitif. Ses bassins d'attraction sont lisses ; dans le cas où le système est algébrique, ses bassins d'attraction sont des ensembles semi-algébriques. Si $F(X) = 0$ est le système à résoudre, et X_0 une approximation initiale, l'homotopie H est définie par : $H(X, t) = tF(X) + (1 - t)(F(X) - F(X_0)) = 0$. L'homotopie est donc une interpolation linéaire entre deux systèmes, d'une part le système $H(X, 0) = F(X) - F(X_0)$ pour $t = 0$, qui s'annule en $X = X_0$, d'autre part le système $H(X, 1) = 0$ pour $t = 1$, qui est le système à résoudre. Le système d'équations $H(X, t) = 0$ décrit une courbe ; la méthode de résolution par homotopie suit cette courbe, soit par une méthode de prédiction et correction, soit par une méthode de simplexe marcheur (*ie* par approximation linéaire par morceau), depuis $t = 0$ et $X = X_0$ jusqu'à $t = 1$. S'arrêter un peu avant $t = 1$ peut éviter des difficultés quand la racine de $F(X) = 0$ est singulière (sa jacobienne n'est pas inversible) ou quand cette racine part à l'infini comme dans : $xy = 1, y = 0$. Il se peut aussi que la courbe suivie n'atteigne pas $t = 1$, mais "redescende" (en considérant la coordonnée t comme une altitude) ; plusieurs topologies possibles pour les courbes d'homotopie sont montrées à la figure 1.13.

Quand le système à résoudre $F(X) = 0$ est algébrique, il est possible de trouver toutes les racines par homotopie ; l'idée est de construire un système initial facile à résoudre, et de même complexité algébrique que F. Par exemple, si F est l'intersection de 2 coniques dans le plan, le système initial peut décrire l'intersection de la conique formée de deux droites ($x(x - 1) = 0$ ou autre) et d'une autre conique formée de deux droites ($y(y - 1) = 0$ ou autre). La résolution de 4 systèmes linéaires donne les 4 racines du système initial ; ces 4 racines sont suivies jusqu'aux 4 racines du système

F. En fait il vaut mieux prendre des racines initiales dans les complexes, ou bien utiliser $H(X,t) = tF(X) + \alpha(1-t)I(X)$ où I est le système initial, et α un nombre complexe de module 1 aléatoire ; il n'y a alors qu'un nombre fini de α pour lesquels la méthode peut échouer, autrement dit pour lesquels la courbe d'homotopie contient un point singulier. Bien sûr, il faut suivre les courbes dans l'espace complexe et non plus dans l'espace réel. Là aussi, arrêter de suivre la courbe un peu avant $t = 1$ est un moyen simple d'éviter les difficultés dues aux solutions qui vont à l'infini et aux solutions singulières du système à résoudre en $t = 1$.

1.5.3. *Arithmétique d'intervalles*

L'analyse par intervalles est de plus en plus utilisée en informatique graphique, par exemple pour trianguler des surfaces algébriques implicites ou segmenter des courbes algébriques implicites. Elle calcule sur des encadrements, représentées par deux nombres flottants, le minimum et le maximum de l'intervalle. Les opérations élémentaires sont décrites ainsi :

$$
\begin{aligned}
&[a,b] + [a',b'] = [\lfloor a + a' \rfloor, \lceil b + b' \rceil] \\
&[a,b] - [a',b'] = [\lfloor a - b' \rfloor, \lceil b - a' \rceil] \\
&[a,b] \times [a',b'] = [\lfloor \min(aa', ab', ba', bb') \rfloor, \lceil \max(aa', ab', ba', bb') \rceil] \\
&1/[a,b] = [\lfloor 1/b \rfloor, \lceil 1/a \rceil] \text{ si } 0 \notin [a,b]
\end{aligned}
$$

où $\lfloor x \rfloor$ désigne le nombre flottant immédiatement inférieur au nombre flottant x, et $\lceil x \rceil$ désigne le nombre flottant immédiatement supérieur : les opérations flottantes doivent être arrondies vers $+\infty$ ou $-\infty$ pour être correctes. Cette arithmétique sur les intervalles ne constitue pas un corps, au sens mathématique ; par exemple un intervalle n'a ni opposé ($[0,1] - [0,1] = [0,1] + (-[0,1]) = [0,1] + [-1,0] = [-1,0]$) ni inverse. L'amplitude d'une somme (ou d'une différence) de deux intervalles est en effet la somme des amplitudes des deux intervalles arguments, et même un peu plus en tenant compte des erreurs d'arrondi. Cette arithmétique se généralise aux fonctions classiques ($\sqrt{x}$, e^x, $\log x$, $\cos x$, $\sin x$...) en décomposant les intervalles des arguments en domaines où la fonction est monotone. Il est aussi possible de calculer des encadrements pour les polynômes, avec la méthode de Horner par exemple. L'arithmétique d'intervalles donne malheureusement des encadrements très pessimistes. Ainsi $x(1-x)$ sur $[0,1]$ égale $[0,1] \times [1,0] = [0,1]$ alors que la réponse exacte est $[0, \frac{1}{4}]$. En fait ce dernier encadrement est l'intervalle optimal pour $y(1-x)$ avec $x \in [0,1], y \in [0,1]$; selon la formule classique, l'arithmétique d'intervalles perd la dépendance entre les variables. Il convient donc d'effectuer certains calculs de façon symbolique pour que $x - x = 0$, ou d'utiliser la forme centrée, ou de Taylor, qui donne des encadrements plus fins, comme dans :

$$f(x_0 \pm \Delta_x, y_0 \pm \Delta_y) \in f(x_0, y_0) + \Delta_x f'_x(x_0 \pm \Delta_x, y_0 \pm \Delta_y) + \Delta_y f'_y(x_0 \pm \Delta_x, y_0 \pm \Delta_y)$$

1.5.4. *Réduction d'intervalles*

Une méthode simple considère les équations comme des règles de réécriture sur des intervalles ; tout système algébrique peut être exprimé par des équations élémentaires telles que : $z = x + y$, $z = xy$, $z = x^2$, en binarisant les expressions ; des équations telles que $z = \cos x$ prennent en compte les fonctions transcendantes. Par exemple, une équation $z = x + y$ donne les règles $Z \leftarrow Z \cap (X + Y)$, $X \leftarrow X \cap (Z - Y)$, $Y \leftarrow Y \cap (Z - X)$, où X, Y, Z désignent les intervalles contenant les inconnues x, y, z, et $\leftarrow$ désigne l'affectation de la variable gauche. Ces affectations sont itérées (par exemple dans un ordre aléatoire) jusqu'à atteindre un point fixe, où l'amplitude des intervalles ne décroît plus de façon significative. Un des intervalles (par exemple le plus large) est alors coupé en deux, et le processus reprend. Luc Jaulin [JAU 00] a utilisé cette méthode de réduction en robotique. Elle est aussi utilisable comme un prétraitement pour une méthode plus sophistiquée.

1.5.5. *Méthode de Krawczyk-Moore*

L'analyse par intervalles a étendu la méthode de la sécante aux intervalles. La méthode de la sécante itère jusqu'à convergence : $x \leftarrow S(x) = x - f(x)J^{-1}$ où J est une approximation de $f'(x)$.

Soit B un vecteur d'intervalles, *ie* une boite, dans laquelle sont cherchées les racines de $f(x) = 0$. J est le jacobien au milieu m de B. Une boite englobant $S(B)$, et aussi appelée $S(B)$, est calculée par intervalles. Si $S(B) = B - f(B)J^{-1}$ est calculée par l'arithmétique d'intervalles naïve, alors l'amplitude de $S(B)$ sera toujours plus grande que celle de B, si bien que $S(B)$ ne sera jamais incluse dans B. Aussi le calcul centré est-il utilisé pour évaluer $S(B)$: ce calcul de $S(B)$ est appelé l'opérateur de Krawczyk, ou de Krawczyk-Moore, selon les auteurs.

Si $S(B) \subset B$, alors B contient un seul point fixe de S, et la méthode de la sécante le trouvera en partant d'un point quelconque de B, par exemple son milieu m. Si $S(B) \cap B = \emptyset$, alors B ne contient pas de racine. Si $S(B) \cap B \neq \emptyset$, alors $S(B) \cap B$ contient éventuellement une solution ou plusieurs : $B \leftarrow S(B) \cap B$ est itéré si $S(B)$ est nettement plus petit que B, sinon B est subdivisé. Plusieurs critères pour choisir quelle variable subdiviser ont été proposés, car il est exclu en haute dimension de diviser en 2 selon toutes les variables. La bissection a un coût exponentiel et il faut donc en user à bon escient ; l'idéal est que chaque bissection sépare des racines.

Toutes les méthodes par intervalles rendent typiquement une liste de boîtes contenant les solutions régulières, et une liste de boites où la méthode ne peut conclure : de telles boites résiduelles contiennent des solutions singulières, "presque singulières", voire pas de solution du tout. La convergence de ces méthodes est ralentie autour des solutions singulières (où le jacobien a un déterminant nul) ou presque singulières.

1.5.6. *Bibliographie*

Neumaier [NEU 90], Kearfott[KEA 96a, KEA 96b], l'équipe COCONUT [BLI 01] ont écrit plusieurs livres sur l'analyse par intervalles. Stolfi et al [AND 94], et Figueiredo et al [FIG 95] ont proposé l'arithmétique d'intervalles affine pour limiter la croissance de l'amplitude des intervalles. Luc Jaulin [JAU 00] a utilisé l'analyse par intervalles en robotique. Stolte et al[STO 01] l'ont utilisé pour discrétiser des surfaces algébriques implicites ou des arbres CSG. Michelucci [MIC 01] l'a utilisé pour couvrir des attracteurs étranges (Julia, Hénon). Gardan et al ont utilisé les bases de Bernstein pour calculer des encadrements de fonctions polynomiales et pour trouver les racines réelles des polynômes en une seule variable. Garloff[GAR 01], puis Mourrain et al[MOU 05], ont utilisé les bases tensorielles de Bernstein pour résoudre des systèmes polynomiaux. Martin et al [MAR 02] ont comparé diverses arithmétiques d'intervalles pour afficher des courbes algébriques implicites.

1.6. Conclusion

Plusieurs questions n'ont pu être détaillées dans cette introduction.

Utiliser des coordonnées pour traduire les contraintes géométriques en équations est la méthode la plus utilisée mais elle introduit des artefacts regrettables ; par exemple un système rigide a $d(d+1)/2$ inconnues de plus que d'équations, en dimension d. Ceci complique les méthodes combinatoires de décompostion. Lu Yang a proposé des équations sans coordonnées ; les inconnues (des distances, des produits scalaires, etc) sont toutes invariantes par changement de repère. Il parvient à résoudre symboliquement certains systèmes, grâce à cette formulation. Cependant, on ne sait pas formuler toutes les contraintes dans ce formalisme, notamment celles relatives aux droites ou aux surfaces en 3D.

Chapitre 2

Bibliographie

[ALD 88] ALDEFELD B., « Variations of geometries based on a geometric-reasoning method », *Computer-Aided Design*, vol. 20, n°3, p. 117-126, Butterworth-Heinemann, 1988.

[AND 94] ANDRADE M. V. A., COMBA J. L. D., STOLFI J., « Affine Arithmetic », *Interval'94*, St Petersburg, 1994.

[BLI 98] BLIEK C., NEVEU B., TROMBETTONI G., « Using Graph Decomposition for Solving Continuous CSPs », *Proc. of Constraint Programming, CP'98*, vol. LNCS 1520, p. 102–116, 1998.

[BLI 01] BLIEK C., SPELLUCCI P., VINCENTE L., NEUMAIER A., GRANVILLIERS L., HUENS E., HENTENRYCK P. V., SAM-HAROUD D., FALTINGS B., *Algorithms for Solving Nonlinear Constrained and Optimisation Problems : State of the Art.*, A 222 page progress report of the COCONUT project, 2001, Available at `http ://www.mat.univie.ac.at/~neum/glopt/coconut/cocbib.html`.

[BOU 95] BOUMA W., FUDOS I., HOFFMANN C., CAI J., PAIGE R., « Geometric constraint solver », *Computer Aided Design*, vol. 27, n°6, p. 487-501, 1995.

[BUT 79] BUTHION M., « Un programme qui résoud formellement des problèmes de construction géométriques », *RAIRO*, vol. 13, n°1, p. 73-106, 1979.

[CHO 88] CHOU S.-C., *Mechanical Geometry theorem Proving*, D. Reidel Publishing Company, 1988.

[DUF 94] DUFOURD J.-F., SCHRECK P., « Un système à base de connaissances pour les constructions géométriques », *Actes de la Conférence AFCET-RFIA, Paris*, p. 351-361, 1994.

[FIG 95] DE FIGUEIREDO L., STOLFI J., « Adaptive Enumeration of Implicit Surfaces with Affine Arithmetic », *Proc. Eurographics Workshop on Implicit Surfaces*, INRIA, p. 161–170, 1995.

[FOU 05] FOUFOU S., MICHELUCCI D., JURZAK J.-P., « Numerical Decomposition of Geometric Constraints », *Proc. of Solid and Physical Modeling, SPM*, p. 143–151, 2005.

[FUD 97] FUDOS I., HOFFMANN C., « A Graph-Constructive Approach to Solving Systems of Geometric Constraints », *ACM Transactions on Graphics*, vol. 16, n°2, p. 179-216, 1997.

[GAO 03] GAO X.-S., ZHANG G.-F., Geometric Constraint Solving Based on Connectivity of Graph, Rapport n°22, Academia Sinica, Beijing, China, december 2003.

[GAR 01] GARLOFF J., SMITH A. P., « Investigation of a subdivision based algorithm for solving systems of polynomial equations », *Journal of nonlinear analysis : Series A Theory and Methods*, vol. 47, n°1, p. 167–178, 2001.

[HEN 92] HENDRICKSON B., « Conditions for unique realizations », *SIAM journal of Computing*, vol. 21, n°1, p. 65-84, 1992.

[HOF 95] HOFFMANN C. M., VERMEER P. J., « A Spatial Constraint Problem », MERLET J.-P., RAVANI B., Eds., *Computational Kinematics'95*, p. 83-92, Kluwer Academic PUBLISHERs, 1995.

[HOF 98] HOFFMANN C., LOMONOSOV A., SITHARAM M., « Geometric Constraint Decomposition », BRUDERLIN B., ROLLER D., Eds., *Geometric Constraint Solving and Applications*, p. 170-195, Springer, 1998.

[HOF 01] HOFFMANN C., LOMONOSOV A., SITHARAM M., « Decomposition of Geometric Constraints Part I : performance measures & Part II : new algorithms », *J. of Symbolic Computation*, vol. 31, n°4, 2001.

[HSU 97] HSU C., BRÜDERLIN B., « A Degree-of-Freedom Graph Approach », *Geometric Modeling : Theory And Practice*, p. 132–155, , 1997.

[JAU 00] JAULIN L., Le calcul ensembliste par analyse par intervalles, Habilitation à diriger des recherches, Université Paris-Sud, Orsay, France, 2000, Available at : http ://www.istia.univ-angers.fr/~jaulin/hdrjaulin.zip.

[JER 02] JERMANN C., Résolution de contraintes géométriques par rigidification récursive et propagation d'intervalles, Ph.D. Thesis, UNSA, NICE, 2002.

[JER 06] JERMANN C., TROMBETTONI G., NEVEU B., MATHIS P., « Decomposition of Geometric Constraint Systems : a Survey », *International journal of computational geometryc and applications – special issue on geometric constraints*, vol. to appear, 2006.

[J.G 93] J. GRAVER B. SERVATIUS H. S., *Combinatorial Rigidity. Graduate Studies in Mathematics*, American Mathematical Society, 1993.

[JOA 03] JOAN-ARINYO R., SOTO-RIERA A., VILA-MARTA S., VILAPLANA-PASTO J., « Transforming an under-constrained geometric constraint problem into a well-constrained one », *SM'03 : Proc. of Solid modeling and applications*, 2003.

[JOA 04] JOAN-ARINYO R., SOTO-RIERA A., VILA-MARTA S., VILAPLANA-PASTO J., « Revisiting decomposition analysis of geometric constraint graphs », *Computer Aided Design*, vol. 36, p. 123–140, 2004.

[KEA 96a] KEARFOTT R. B., KREINOVICH V., Eds., *Applications of Interval Computations*, Kluwer, Dordrecht, the Netherlands, 1996.

[KEA 96b] KEARFOTT R., *Rigorous Global Search : Continuous Problems*, Kluwer, Dordrecht, Netherlands, 1996.

[LAM 98] LAMURE H., MICHELUCCI D., « Qualitative study of geometric constraints », *Geometric Constraint Solving and Applications*, p. 234–258, Springer, 1998.

[LAT 96] LATHAM R., MIDDLEDITCH A., « Connectivity Analysis : A Tool For Processing Geometric Constraints », *Computer Aided Design*, vol. 28, n°11, p. 917-928, 1996.

[MAR 71] MARTIN W., « Determining the Equivalence of Algebraic Expressions by Hash Coding », *J. ACM*, vol. 18, n°4, p. 549–558, 1971.

[MAR 02] MARTIN R., SHOU H., VOICULESCU I., BOWYER A., WANG G., « Comparison of interval methods for plotting algebraic curves », *Comput. Aided Geom. Des.*, vol. 19, n°7, p. 553–587, Elsevier Science Publishers B. V., 2002.

[MIC 01] MICHELUCCI D., « *Reliable representations of strange attractors* », p. 379–389, Kluwer, 2001.

[MOU 05] MOURRAIN B., ROUILLIER F., ROY M.-F., « Bernstein's basis and real root isolation », *Combinatorial and Computational Geometry*, Mathematical Sciences Research Institute Publications, Cambridge University Press, 2005, to appear.

[NEU 90] NEUMAIER A., *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge, UK, 1990.

[OWE 91] OWEN J., « Algebraic Solution For Geometry From Dimensional Constraints », *Proc. of Solid Modeling and CAD/CAM Applications*, p. 397-407, 1991.

[SCH 80] SCHWARTZ J., « Fast Probabilistic Algorithms for Verification of Polynomial Identities », *J. ACM*, vol. 4, n°27, p. 701-717, 1980.

[SIT 03] SITHARAM M., Frontier, an opensource gnu geometric constraint solver : version3 (2003) for general 2d and 3d systems, 2003, http ://www.cise.ufl.edu/ sitharam.

[SOS 02] SOSNOV A., MACÉ P., « Rapid Algebraic Resolution of 3D Geometric Constraints and Control of their Consistency », *Proceedings of the 4th International Workshop on Automated Deduction in Geometry*, 2002.

[STO 01] STOLTE N., KAUFMAN A., « Novel Techniques for Robust Voxelization and Visualization of Implicit Surfaces », *Graphical Models*, vol. 6, n°63, p. 387–412, Academic Press, 2001.

[SUN 86] SUNDE G., « Specification of Shapes by Dimensions and Other Geometric Constraints », *IFIP WG 5.2 Geometric Modeling*, 1986.

[SUT 63] SUTHERLAND I., Sketchpad : A Man-Machine Graphical Communication System, PhD thesis, Department of Electrical Engineering, MIT, 1963.

[TRO 97] TROMBETTONI G., Algorithmes de maintien de solution par propagation locales pour les systèmes de contraintes, Ph.D. Thesis, UNSA, NICE, 1997.

[TRO 98] TROMBETTONI G., « A Polynomial Time Local Propagation Algorithm for General Dataflow Constraint Problems », *Proc. of Constraint Programming*, vol. LNCS 1520, p. 432–446, 1998.

[TRO 06] TROMBETTONI G., WILCZKOWIAK M., « GPDOF : a Fast Algorithm to Decompose Under-constrained Geometric Constraints : Application to 3D Model Reconstruction », *Int. Journal of Computational Geometry and Applications (IJCGA)*, vol. 16, 2006, submitted.

[VER 92] VERROUST A., SCHONEK F., ROLLER D., « Rule Oriented Method for Parametrized Computer Aided Design », *Computer Aided Design*, vol. 24, n°6, p. 531–540, 1992.