Journées du Groupe de Travail en Modélisation Géométrique 2014, Lyon

Réduire les irréductibles

Jean-Marc Cane[1], Arnaud Kubicki[1], Dominique Michelucci[1], Hichem Barki[2], Sebti Foufou[2]

[1] LE2I laboratory, Université de Bourgogne, BP 47870, 21078 Dijon France
[2] Dept. of Computer Science and Engineering, CENG, Qatar University, Doha, Qatar

Résumé

You recklessly told your boss that to solve a nonlinear system of size n (n unknowns and n equations) requires a time proportional to n (you were not very attentive during algorithmic complexity lectures). So now, you have only one night to solve a problem with size 1000 *and not to be fired tomorrow morning. The most frustrating thing is that if you knew the values of five key unknowns, then the system would be reducible into small square subsystems and easily solvable. You wonder if it would not be possible to exploit this reducibility, even without knowing the values of key unknowns. This article shows that it is indeed possible. This is done at the lowest level, at the linear algebra routines level, so that numerous solvers (Newton-Raphson, homotopy, and also P-adic method relying on Hensel lifting) can benefit from this decomposition with minor modifications : for example, with $k \ll n$ key unknowns, the cost of a Newton iteration becomes $O(kn^2)$ instead of $O(n^3)$.*

Vous avez imprudemment assuré à votre chef que résoudre un système d'équations non linéaires de taille n (n équations et n inconnues) nécessite un temps proportionnel à n (vous n'étiez pas très assidu aux cours d'algorithmique et complexité). Pour ne pas être licencié demain matin, il ne vous reste plus que cette nuit pour résoudre un système de taille 1000. Le plus frustrant est que, si vous connaissiez la valeur de 5 inconnues clefs, alors le système se décomposerait en de nombreux sous systèmes plus petits, et serait facilement soluble. Vous vous demandez s'il ne serait pas possible d'utiliser cette décomposition, bien que vous ne connaissiez pas les valeurs des inconnues clefs. Cet article montre que c'est possible. Ceci est fait au plus bas niveau, celui des procédures d'algèbre linéaire, si bien que de nombreux solveurs peuvent bénéficier de cette décomposition. Par exemple, avec $k \ll n$ inconnues clefs, le coût d'une itération de la méthode de Newton-Raphson chute de $O(n^3)$ à $O(kn^2)$.

Mots clefs. Modélisation géométrique par contraintes, résolution de contraintes géométriques, re-paramétrisation, algèbre linéaire, réduction, décomposition, couplage maximum, graphe biparti, Dulmage-Mendelsohn, König-Hall.

1. Introduction

La modélisation géométrique par contraintes [BR98, HJA05, Hof06, BH11, JTNM06] nécessite la résolution de gros systèmes d'équations non linéaires (souvent algébriques). L'article [AAJM93] se fonde sur les propriétés des graphes bipartis sous jacents aux systèmes d'équations pour décomposer en temps polynomial les systèmes en parties sur-, sous-, ou bien-contraintes ; cet article donne aussi une méthode efficace pour décomposer les systèmes bien contraints en sous systèmes bien contraints et irréductibles. Ces décompositions accélèrent grandement la résolution. Elles permettent aussi de détecter des erreurs dans les systèmes de contraintes : la programmation par contraintes est de la programmation... Toutefois les méthodes de cet article ont des limitations ; par exemple, elles ne s'appliquent pas aux systèmes re-paramétrés, proposés dans [GHY04, GHY02, FS08, IMS11, MSI12]. La re-paramétrisation détecte ou introduit des inconnues clefs, aussi appelées paramètres (d'où le terme de re-paramétrisation) qui ont la propriété suivante : si leurs valeurs étaient connues, alors le système serait décomposable et facilement soluble.

Cet article montre qu'il est possible de bénéficier de cette décomposabilité, et ce même quand les valeurs des inconnues clefs sont inconnues. Il propose d'utiliser la re-paramétrisation au niveau le plus bas, celui des procédures d'algèbre linéaire, pour la résolution de systèmes linéaires ou l'inversion de matrices. De nombreux solveurs (Newton-Raphson, homotopie) utilisent de telles procédures. Ce ni-

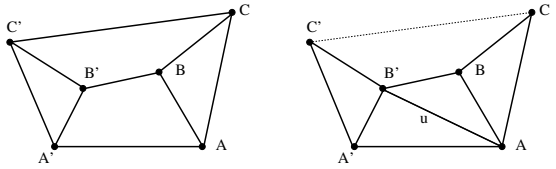

Figure 1: *A gauche : un système de contraintes géoméytriques en 2D. A droite : le système après re-paramétrisation : si $u = AB'$ était connu, alors le système serait réductible et facile à résoudre ; l'équation fixant la longueur CC' serait redondante et pourrait être ignorée.*

veau semble donc être le meilleur (du moins pour les plus paresseux d'entre nous) pour tirer avantage de la re-paramétrisation au moindre coût. Ceci n'empêche d'ailleurs pas de tenter de profiter de la re-paramétrisation à un niveau plus élevé dans les algorithmes.

§2 présente la re-paramétrisation sur des exemples. §3 présente la théorie du couplage dans les graphes bipartis : les méthodes de décomposition s'appuient sur cette théorie. §4 montre comment la décomposition en sous systèmes accélère les procédures d'algèbre linéaires. §5 montre comment la re-paramétrisation accélère les procédures d'algèbre linéaire pour les systèmes re-paramétrés ; ainsi la méthode de Newton ou l'homotopie peuvent tirer parti de la re-paramétrisation très facilement. §6 présente la complexité algorithmique de cette méthode. §7 explique que la remontée de Hensel dans les méthodes P-adiques peut aussi profiter de la re-paramétrisation. C'est important car cela signifie que ce n'est pas seulement l'analyse numérique qui peut bénéficier de la re-paramétrisation, mais aussi le calcul symbolique. §8 montre que les solveurs par intervalles devraient aussi pouvoir tirer parti de la re-paramétrisation ; cependant l'effet enveloppant nécessite d'autres recherches. §9 envisage brièvement l'usage de la re-paramétrisation à un niveau supérieur. §10 présente les travaux futurs et les questions soulevées. §11 conclut.

2. La re-paramétrisation

2.1. Un exemple 2D simple

La Fig.1 montre un système de contraintes géométriques en 2D. Les longueurs des arêtes sont spécifiées. Il est facile de calculer les coordonnées 2D des sommets de l'un des triangles, disons $A'B'C'$, et de fixer dans le plan ce triangle (par exemple en fixant $x'_A = y'_A = y'_B = 0$). Ensuite il est facile de trouver x'_B, x'_C, y'_C. Il reste alors 6 inconnues $x_A, y_A, x_B, y_B, x_C, y_C$: ce sont les coordonnées des points A, B, C, et il reste 6 équations, fixant les distances AB, BC, CA et AA', BB', CC'. Ce système de 6 équations et 6 inconnues est bien contraint : il a un nombre fini de solutions réelles (du moins pour des valeurs suffisament génériques des longueurs –par exemple non nulles). Malheureusement, ce système est irréductible : il ne peut pas être décomposé en sous systèmes bien contraints plus petits. Nous remarquons que, si la longueur u de l'arête AB' était connue (elle ne l'est pas), alors il serait facile de calculer les coordonnées de A, en résolvant un système de 2 équations à 2 inconnues x_A, y_A ; géométriquement, A est l'intersection de deux cercles, l'un de centre A' et de rayon AA', le second de centre B' et de rayon u. Ensuite les coordonnées de B pourraient être calculées de la même façon, comme intersection de deux cercles de centres et de rayons connus. Finalement le point C pourrait être calculé, de trois façons différentes ; en effet deux équations suffisent pour fixer C, et trois sont disponibles. Une de ces équations pourrait être supprimée, toujours en supposant que la valeur de u est connue. Disons que l'équation redondante, "oubliable", est celle fixant la longueur de CC'. Pour récapituler : si u était connu, le système serait décomposable et facilement résolu.

Mais, en fait, la valeur de u est inconnue. Pour trouver cette valeur, il est possible de tracer ou d'échantillonner la fonction $A(u), B(u), C(u)$, et de détecter quand l'équation oubliée (celle spécifiant la longueur CC') est satisfaite. Quelques itérations de Newton, ou de la dichotomie, précisent la valeur du u solution. En un sens, la re-paramétrisation remplace le système initial de 6 équations et 6 inconnues en un système à une seule équation (l'équation oubliée) et une seule inconnue (u) ; il est donc logique que la résolution soit accélérée. Mais plusieurs complications surgissent. Elles sont dues au fait que $A(u), B(u), C(u)$ sont des multi-fonctions (algébriquement, des racines carrées apparaissent ; géométriquement, deux cercles peuvent avoir 2 points d'intersection), et au fait que la construction de $A(u), B(u), C(u)$ peut échouer pour certaines valeurs de u (algébriquement pour la racine carrée d'un nombre négatif ; géométriquement, pour l'intersection de deux cercles disjoints). Enfin, comment choisir l'intervalle des valeurs pour u ? Pour cet exemple, l'inégalité triangulaire fournit des intervalles ; dans le triangle $AA'B'$, l'inconnue clef u ne peut pas être plus grande que $AA' + A'B'$ et elle ne peut pas être plus petite que $|AA' - A'B'|$; de même, avec le triangle ABB', l'inconnue clef u ne peut pas être plus grande que $AB + BB'$ et ne peut pas être plus petite que $|AB - BB'|$. Quand ces deux intervalles sont disjoints, il n'y a pas de solution réelle.

Définissons le grand système re-paramétré comme la concaténation du système initial et de l'équation $u^2 - (A - B') \cdot (A - B') = 0$, qui définit l'inconnue clef u. Le grand système a une équation et une inconnue de plus que le système initial à 6 équations et 6 inconnues. Le grand système est lui aussi bien contraint, *i.e.,* il a autant d'équations que d'inconnues ; ces équations sont indépendantes (le jacobien est de plein rang presque partout), donc le grand système a un nombre fini de solutions réelles (au moins pour des valeurs génériques des longueurs).

La valeur de u est inconnue, et le grand système est irré-

ductible. En effet ses équations sont :

$$0 = \text{dist}^2(A,B')^2 - u^2 \quad \text{définition de } u \tag{1}$$
$$0 = \text{dist}^2(A',A) - l_{A'A}^2 \tag{2}$$
$$0 = \text{dist}^2(B',B) - l_{B'B}^2 \tag{3}$$
$$0 = \text{dist}^2(A,B) - l_{AB}^2 \tag{4}$$
$$0 = \text{dist}^2(A,C) - l_{AC}^2 \tag{5}$$
$$0 = \text{dist}^2(B,C) - l_{BC}^2 \tag{6}$$
$$0 = \text{dist}^2(C',C) - l_{C'C}^2 \quad \text{équation oubliée} \tag{7}$$

où $l_{A'A}$, $l_{B'B}$, l_{AB}, l_{AC}, l_{BC}, $l_{C'C}$ sont les six longueurs spécifiées, et $\text{dist}^2(A,B)^2 = (x_A - x_B)^2 + (y_A - y_B)^2$. Le jacobien a la structure suivante (une étoile représente une valeur non nulle) :

u	x_A	y_A	x_B	y_B	x_C	y_C
$*$	$*$	$*$	0	0	0	0
0	$*$	$*$	0	0	0	0
0	0	0	$*$	$*$	0	0
0	$*$	$*$	$*$	$*$	0	0
0	$*$	$*$	0	0	$*$	$*$
0	0	0	$*$	$*$	$*$	$*$
0	0	0	0	0	$*$	$*$

(8)

Si la première colonne (pour u) et la dernière ligne (le gradient de l'équation oubliée) étaient supprimées, *i.e.,* si la valeur de u était connue, rendant redondante et inutile la dernière équation, le système restant serait réductible. L'ordre des équations et des inconnues a été choisi pour rendre visible la décomposition en trois blocs de 2 équations chacun, avec une structure triangulaire inférieure par blocs.

D'où la question traitée dans cet article : est il possible d'utiliser la re-paramétrisation, *i.e.,* est il possible de réduire d'une certaine façon le grand système, bien que la valeur de l'inconnue clef soit, justement, inconnue ? A première vue, cela semble impossible.

2.2. Un exemple 3D, l'hexaèdre

Le principe de la re-paramétrisation est illustré avec le problème 3D de l'hexaèdre (une généralisation du cube), Fig.2. Les longueurs des 12 arêtes sont fixées ; le système de contraintes est complété par 6 contraintes de coplanarité de 4 points, une contrainte par face, et une contrainte de non dégénérescence : l'hexaèdre ne doit pas être aplati (cette dernière contrainte élimine un continuum de solutions dégénérées, une "courbe" d'hexaèdres aplatis). Ce système a un nombre fini de solutions réelles, modulo les déplacements. Il a 18 équations. Les 8 sommets sont représentés par 3 fois 8, soit 24 coordonnées inconnues, dont six peuvent être fixées, comme d'habitude : par exemple le sommet 0 est fixé à l'origine, le sommet 1 est fixé sur l'axe des x, le sommet 2 est fixé dans le plan Oxy. Donc $x_0 = y_0 = z_0 = y_1 = z_1 = z_2 = 0$. Il reste un système de 18 équations et inconnues. Puis nous

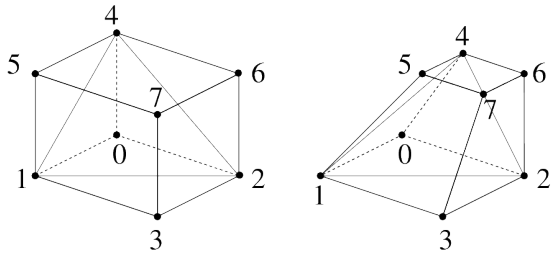

Figure 2: *Deux hexaèdres. Les longueurs des 12 arêtes (en gras) sont données. Les longueurs des trois diagonales (le triangle 124) sont les inconnues clefs. Si leurs valeurs étaient connues, le système serait réductible et facilement résolu. Les 3 contraintes de longueur des arêtes incidentes au sommet 7 seraient redondantes.*

voyons que $z_3 = 0$ (à cause de la co-planarité des sommets 0, 1, 2, 4) et $x_1 = l_{01}$ (où l_{01} est la longueur spécifiée pour l'arête 01), ce qui satisfait deux contraintes que nous pouvons éliminer. Il nous reste à résoudre un système de 16 équations et 16 inconnues. Ce dernier est irréductible, selon les méthodes de l'article [AAJM93].

L'idée de la re-paramétrisation est comme suit : supposons connues les longueurs des arêtes du triangle 1, 2, 4. Alors il est possible de calculer les coordonnées du sommet 3, en fonction des coordonnées des sommets 0, 1, 2. De même, il est possible de calculer les coordonnées du sommet 5, en fonction des coordonnées des sommets 0, 1, 4. De même, il est possible de calculer les coordonnées du sommet 6, en fonction des coordonnées des sommets 0, 2, 4. Enfin, les équations des plans 135, 236 et 456 peuvent être calculées. Leur point d'intersection est le sommet numéro 7.

Les trois contraintes de longueur des arêtes 37, 57, 67 n'ont pas été utilisées : elles sont redondantes. Dans cet exemple, les inconnues clefs ne sont pas des inconnues du système initial ; ce sont les longueurs du triangle 124 ; les trois contraintes de longueurs des arêtes $37, 57, 67$ sont les contraintes redondantes ou "oubliées".

En un sens, la re-paramétrisation change le système initial $S(X) = 0$ en un système : $X = F(U), G(X) = G(F(U)) = 0$, où les U sont les inconnues clefs, où $X = F(U)$ garantit que les contraintes non redondantes seront satisfaites, par construction, et où $G(F(U)) = 0$ sont les équations redondantes. D'une certaine façon, les inconnues du petit système re-paramétré sont les inconnues clefs U, et ses équations sont $G(F(U)) = 0$: les inconnues X n'apparaissent plus dans cette formulation. Une difficulté apparait cependant : F est une multi-fonction (par exemple, elle utilise $\pm\sqrt{}$), et elle peut être non définie pour certaines valeurs de U. Il peut aussi être malcommode ou difficile d'expliciter la fonction F. Pour ces raisons, il est préférable d'utiliser la formulation implicte : $F(U,X) = 0, G(X) = 0$, ou même $F(U,X) = 0, G(U,X) = 0$, qui est la plus générale.

Quand il n'y a qu'un seule inconnue clef dans U, calculer la valeur solution U revient à suivre une courbe paramétrée $X = F(U)$ ou une courbe d'équation implicite $F(U,X) = 0$, et à détecter quand la contrainte oubliée est satisfaite (par exemple, dans le cas paramétré, $G(U)$ change de signe) ; quelques itérations de Newton, ou quelques dichotomies, précisent alors la valeur solution de U.

Quand il y a plus d'une inconnue clef, les méthodes proposées dans la littérature pour résoudre en U deviennent pour le moins malcommodes. Cet article présente une méthode simple, utilisable pour un nombre quelconque d'inconnues clefs ; la seule restriction est que le nombre d'inconnues clefs, k, doit rester petit devant n, la taille du système initial.

3. La théorie du couplage

Chaque itération de la méthode de Newton-Raphson inverse une matrice jacobienne, ou bien résout des systèmes linéaires, ou bien calcule une décomposition LU, bref effectue des calculs d'algèbre linéaire. C'est aussi le cas pour d'autres solveurs, comme l'homotopie (aussi appelée continuation).

Les méthodes de [AAJM93] réduisent les systèmes (linéaires, ou non linéaires) bien contraints en sous systèmes irréductibles et bien contraints. Ces méthodes sont uniquement combinatoires. Elles étudient le graphe biparti des équations et des inconnues du système. Chaque équation est représentée par un sommet, de même que chaque inconnue. Une arête relie le sommet d'une équation à un sommet d'une inconnue ssi l'équation dépend de l'inconnue. Ce sont les seules arêtes du graphe, qui est donc biparti : un premier ensemble de sommets est constitué des sommets des équations, et le deuxième ensemble de sommets est constitué des sommets des inconnues.

Ces méthodes s'appliquent aussi bien sur les systèmes linéaires que non linéaires. Elles utilisent les couplages maximum. Un *couplage* est un sous ensemble des arêtes du graphe, tel que tout sommet appartient à au plus une arête du couplage. Un sommet qui appartient à une des arêtes du couplage est dit *saturé* ou *couvert* par le couplage ; on dit que le couplage *couvre*, ou *sature*, ce sommet. Un couplage est maximum, si sa cardinalité (son nombre d'arêtes) est maximum. Un couplage est *maximal* s'il est maximal pour l'inclusion : il n'est pas inclus dans un autre couplage. Un couplage maximal n'est pas toujours maximum, mais tout couplage maximum est bien sûr maximal. Un couplage est *parfait* si tous les sommets sont saturés. Un couplage parfait est maximum et maximal.

Les propriétés *structurelles* d'un système d'équations, ou de sa jacobienne, sont valables, avec probabilité un, pour toutes les valeurs des coefficients de ce système ; elles ne dépendent que du graphe biparti associé. Par exemple, une matrice est de rang plein, ou un système est bien contraint, ssi le

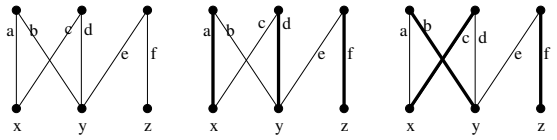

Figure 3: *Le graphe biparti d'un système linéaire $ax+by = r_1, cx+dy = r_2, ey+fz = r_3$, et ses deux couplages parfaits adf et bcf. Le déterminant de la matrice sous jacente est $adf - bcf$.*

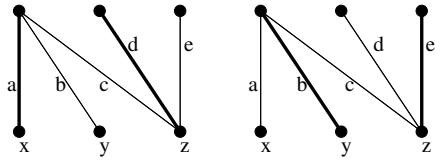

Figure 4: *Le système $F_1(x,y,z) = F_2(z) = F_3(z) = 0$ est structurellement mal contraint. "Structurellement" signifie que le système est mal contraint quelles que soient les valeurs de a,b,c,d,e. Aussi son graphe biparti n'a-t-il aucun couplage parfait. Les couplages maximum contiennent deux arêtes (donc le rang de la matrice est 2, pour des valeurs génériques des coefficients a,b,c,d,e). Ce sont (a,d), (a,e), (b,d) et (b,e).*

graphe biparti associé a (au moins) un couplage parfait. Plus généralement, le rang d'une matrice, ou le nombre d'équations indépendantes d'un système d'équations, ne peut pas être plus grand que le nombre d'arêtes d'un couplage maximum du graphe biparti associé. En général, le rang et le cardinal des couplages maximum sont égaux. Dans les cas particuliers, ou dégénérés, le rang de la matrice (ou du système) est inférieur à la cardinalité des couplages maximums. La bijection entre chaque couplage parfait et chaque terme du déterminant d'une matrice (par exemple jacobienne) est illustrée Fig.3 pour la matrice :

$$M = \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & e & f \end{pmatrix} \tag{9}$$

Elle a pour déterminant $adf - bcf$, et les couplages parfaits sont adf et bcf. Les arêtes du graphe sont étiquetées avec le coefficient correspondant de la matrice. Notez que sur cet exemple, le déterminant ne dépend pas du coefficient e, et l'arête étiquetée e n'appartient à aucun couplage parfait. Cette correspondance entre couplage parfait et terme du déterminant explique pourquoi, lorsqu'il n'y a aucun couplage parfait, le déterminant est identiquement nul, comme pour le système structurellement mal contraint Fig.4. Le mot "structurellement" signifie que le système est mal contraint quelles que soient les valeurs de ses coefficients a,b,c,d,e.

Les méthodes de [AAJM93] calculent en temps fortement polynomial un couplage maximum (parfait pour les

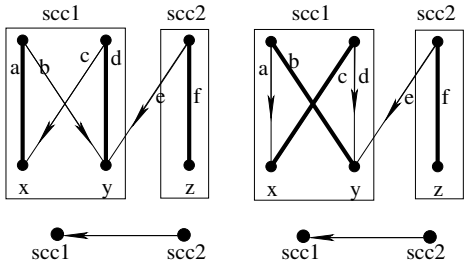

Figure 5: *Au dessus : les deux couplages parfaits du système Fig.3, et les composantes fortement connexes induites dans le graphe orienté ; les arêtes du couplage sont orientées dans les deux sens, et les autres de haut en bas (des équations vers les inconnues). En dessous : dans le graphe réduit, chaque composante fortement connexe est réduite à un sommet. Le graphe réduit donne les dépendances entre les sous systèmes irréductibles. Le graphe réduit est sans cycle.*

sytèmes bien contraints) du graphe biparti associé à un système d'équations. Ensuite elles orientent les arêtes du graphe, selon qu'elles appartiennent ou non au couplage maximum. C'est illustré Fig.5 pour l'exemple de la Fig.3. Dans le cas d'un système structurellement bien contraint (le seul cas discuté dans cet article), chaque composante fortement connexe de ce graphe orienté représente un sous système bien contraint et irréductible ; les composantes fortement connexes sont indépendantes du couplage maximum utilisé. De plus, les arcs entre deux composantes fortement connexes, autrement dit les arcs du graphe réduit (acyclique), donnent les dépendances entre les sous-systèmes, autrement dit l'ordre dans lequel résoudre les sous-systèmes. Le livre de Lovasz et Plummer : "Matching theory" détaille la théorie du couplage ; le lecteur trouvera dans ce livre les liens entre couplage et matroïde, les théorèmes de König-Hall, la décomposition de Dulmage et Mendelsohn, etc.

Certaines structures de données sont non seulement commodes, mais quasiment indispensables, pour profiter pleinement et facilement de la décomposition des systèmes, lors de leur résolution, et cela même pour des systèmes linéaires. Il s'agit des graphes bipartis équations-inconnues, et des DAGs ("Directed Acyclic Graph") de sous-systèmes, dont les arcs indiquent les dépendances entre sous systèmes. Ces DAGs sont les graphes réduits en Fig.5 Ces structures de données sont encore plus indispensables dans le cas des systèmes non linéaires, où il faut gérer la multiplicité des solutions des sous systèmes non linéaires, ou au contraire leur absence de solutions.

4. La réduction accélère les calculs d'algèbre linéaire

Les méthodes de [AAJM93] peuvent ordonner en temps polynomial les inconnues, et les équations, de telle façon que la matrice jacobienne du système devienne triangulaire inférieure. Cette structure rend visible la décomposition en sous systèmes bien contraints irréductibles. Elle accélère la résolution des systèmes linéaires, ou l'inversion de la matrice jacobienne. En effet il devient possible d'utiliser la substitution en-avant, par bloc ("forward block substitution"). Rappelons que la résolution d'un système linéaire de taille n est dans le cas général en $O(n^3)$, alors qu'elle est en $O(n^2)$ pour une matrice triangulaire inférieure [CLRS09].

Cette section suppose que la matrice M est triangulaire inférieure par bloc, comme ceci par exemple :

$$M = \begin{pmatrix} M_{1,1} & 0 & 0 \\ M_{2,1} & M_{2,2} & 0 \\ M_{3,1} & M_{3,2} & M_{3,3} \end{pmatrix} \tag{10}$$

Nous montrons maintenant comment tirer profit de la décomposition de M pour résoudre un système linéaire $MX = B$ ou pour inverser la matrice M. Les preuves sont classiques et omises [CLRS09].

Remarquons que les matrices et vecteurs dans cette section peuvent contenir des nombres flottants (quand utilisés avec la méthode de Newton-Raphson §5), ou des intervalles (quand utilisés par des solveurs par intervalles §8), ou des éléments d'un corps fini ou d'un corps P-adique (§7).

4.1. Inverse d'une matrice triangulaire inférieure par bloc

L'inverse K d'une matrice M carrée triangulaire inférieure par bloc, inversible, est aussi triangulaire inférieure par bloc. M et K ont la même structure par bloc.

$$M^{-1} = K = \begin{pmatrix} K_{1,1} & 0 & 0 \\ K_{2,1} & K_{2,2} & 0 \\ K_{3,1} & K_{3,2} & K_{3,3} \end{pmatrix} \tag{11}$$

Les blocs de $K = M^{-1}$ sont :

$$K_{l,c} = 0 \text{ when } c > l \tag{12}$$

$$K_{l,l} = M_{l,l}^{-1} \tag{13}$$

$$K_{l,c} = -K_{l,l} \sum_{i=c}^{l-1} M_{l,i} K_{i,c} \text{ when } l > c \tag{14}$$

En fait, il est toujours possible d'éviter l'inversion de la matrice M : il suffit de résoudre moins de n systèmes linéaires (n étant le nombre de lignes, ou de colonnes, de M) comme nous le verrons dans la suite.

4.2. Résoudre un système linéaire

Pour résoudre $MX = B$, où $X = (X_1, X_2, \ldots)^t$, et $B = (B_1, B_2, \ldots)^t$ ont une structure par bloc compatible avec celle de M, nous résolvons $X_1 = \text{solve}(M_{1,1}, B_1)$, puis

$X_2 = \text{solve}(M_{2,2}, B_2 - M_{2,1}X_1)$, puis $X_3 = \text{solve}(M_{3,3}, B_3 - (M_{3,1}X_1 + M_{3,2}X_2))$, etc, *i.e.,* nous résolvons $X_k = \text{solve}(M_{k,k}, B_k - \sum_{i=1}^{k-1} M_{k,i}X_i)$. Plus les blocs $M_{k,k}$ sont petits, plus grande est l'accélération. Par exemple, si M est de taille m par m et tous les blocs diagonaux sont de taille 1, résoudre est en $O(m^2)$ pour une matrice triangulaire inférieure dense (ce qui est optimal, car c'est la taille des données du problème, et la taille du résultat), au lieu de $O(m^3)$ dans le cas général. Bien sûr, la matrice M peut aussi être peu dense ; en fait si elle a $O(\alpha)$ entrées non nulles, et que seuls sont effectués les calculs indispensables, alors résoudre un système linéaire est en $O(\alpha)$. Pour les systèmes de contraintes géométriques, α est souvent de l'ordre de n. L'accélération due à la décomposition peut donc être considérable.

5. La re-paramétrisation accélère les calculs d'algèbre linéaire

Avec la re-paramétrisation, et après la décomposition et le ré-ordonnancement calculés par les méthodes de l'article [AAJM93], chaque itération de Newton-Raphson résout un système linéaire où la matrice a la structure (déjà vue en Eq.8 ou exemple en Fig.1) montrée Fig.6.

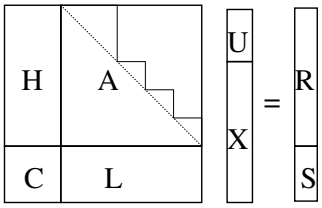

Figure 6: *The structure of the Jacobian of a big re-parameterizd system.*

En formules :

$$HU + AX = R, CU + LX = S \tag{15}$$

où la matrice A est carrée, inversible, triangulaire inférieure par bloc. U est le vecteur des inconnues clefs. X est le vecteur des autres inconnues. La partie $HU + AX = R$ vient de la dérivée des contraintes non oubliées ; la partie $CU + LX = S$ vient de la dérivée des contraintes oubliées. Le lecteur peut comparer Eq.15 et Eq.8.

Définition : cette matrice

$$\begin{pmatrix} H & A \\ C & L \end{pmatrix}$$

a une structure R (R pour re-paramétrisation). Deux matrices ont la même structure R ssi les tailles de leurs blocs H, A, C, L sont égales et les structures de bloc de leur sous matrice A sont égales.

La re-paramétrisation rend A fortement réductible (c'est le but de la re-paramétrisation). Cette décomposition de A permet d'accélérer la résolution des systèmes linéaires $Ax = b$ et l'inversion de A, comme vue à la section précédente.

Donc $X = A^{-1}(R - HU)$ et A^{-1} peuvent être calculées avec la méthode en §4.1. Ensuite, en remplaçant X par sa valeur $A^{-1}(R - HU)$, nous obtenons :

$$CU + LX = S \tag{16}$$

$$\Rightarrow \quad CU + LA^{-1}(R - HU) = S \tag{17}$$

$$\Rightarrow \quad (C - LA^{-1}H)U = S - LA^{-1}R \tag{18}$$

Nous résolvons $(C - LA^{-1}H)U = S - LA^{-1}R$ avec la méthode en §4.2. Remarquons que la matrice carrée inversible $C - LA^{-1}H$ a une taille très petite devant celle de A. Par exemple, pour le problème de l'hexaèdre, A est de taille 16×16, et C et $C - LA^{-1}H$ sont de taille 3×3. Pour le problème 2D en Fig.1, A est de taille 6×6, et C et $C - LA^{-1}H$ sont de taille 1×1.

Comme d'habitude, l'inversion de la matrice (ici A) peut être évitée. La matrice A^{-1} apparait d'abord dans l'expression : $A^{-1}R$; posons $Z = A^{-1}R$. Alors $AZ = R$, ce qui est une équation linéaire, dont le vecteur inconnu est Z ; cette équation est résolue par la méthode de §4.2. La matrice A^{-1} apparait aussi dans l'expression : $A^{-1}H$. De nouveau, posons $Z = A^{-1}H$, donc $AZ = H$. Le nombre de colonnes de H, et de Z, est le nombre d'inconnues clefs, k ; donc résoudre $k \ll n$ systèmes linéaires, un pour chaque colonne de H, nécessite k appels à la méthode en §4.2. k est petit devant n, la taille de A. Ce calcul est donc plus rapide que celui calculant l'inverse de A.

La section suivante étudie le coût de cette méthode, du point de vue de la complexité algorithmique.

6. Complexité

Soit α la complexité de la résolution d'un système linéaire $x := \text{solve}(Ax = b)$ par la méthode en §4.2. Il faut résoudre $k+1$ de ces systèmes linéaires pour calculer $A^{-1}H$ et $A^{-1}R$: ceci coûte $O(k\alpha)$. Ensuite, calculer $L_{k\times n}(A^{-1}R)_{n\times 1}$ coûte $O(kn)$. Résoudre le système linéaire de Eq.18, *i.e.,* calculer U, coûte $O(k^3)$ (comme k est petit devant n, il est inutile d'optimiser ce calcul). Puis nous calculons $X := (A^{-1}R)_{n\times 1} - (A^{-1}H)_{n\times k}U_{k\times 1}$; le vecteur $(A^{-1}R)$ est connu, et la matrice $A^{-1}H$ aussi, donc calculer X coûte $O(kn)$.

Au total, calculer U et X coûte $O(k\alpha + k^2n + k^3)$, ou $O(k(\alpha + kn + k^2))$.α dépend de la structure de A, triangulaire inférieure par blocs.

Discutons maintenant l'ordre de grandeur de α. Pour des matrices A peu denses, α peut être de l'ordre de n, et donc la méthode est en $O(k^2(n+k))$; c'est souvent le cas pour les systèmes de contraintes géométriques. Pour A une matrice

triangulaire inférieure dense (tous les blocs sont de taille 1), α est en $O(n^2)$ et la méthode (pour résoudre un système linéaire re-paramétré) est alors en $O(k(n^2+kn+k^2))$. C'est toujours un gain important, comparé au coût en $O(n^3)$ d'une itération de Newton-Raphson dans le cas général. En fait, quand A est triangulaire inférieure, α est juste le nombre de cases structurellement non nulles dans A.

En pratique, A est triangulaire inférieure par bloc. Supposons que le système $Ax = b$ est décomposé en s sous-systèmes de tailles égales, ou à peu près, pour simplifier. Alors α est en $O(s \times (n/s)^3) = O(n^3/s^2)$. Ceci est à comparer avec le coût de résolution d'un système linéaire dans le cas général, qui est $O(n^3)$ (*i.e.,* $s = 1$).

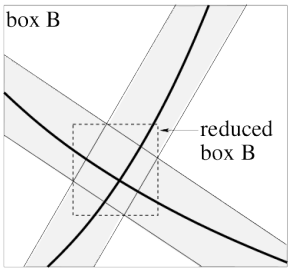

Figure 7: *La linéarisation par intervalles d'un système en 2D dans une boite B. Les courbes sont prises en sandwich dans des traits épais. Résoudre donne la boite réduite en pointillé.*

7. Le cas P-adique

Les méthodes P-adiques sont utilisées en analyse P-adique, mais aussi pour résoudre des problèmes diophantiens ou pour du calcul formel (PGCD de polynômes, factorisation de polynômes).

La remontée de Hensel utilisée dans les méthodes P-adiques peut aussi bénéficier de la re-paramétrisation. Supposons que X_0 est une racine d'un système algébrique $F(X) = 0$ (supposé re-paramétré) modulo p un nombre premier, ou une puissance d'un nombre premier. Nous voulons calculer X_1 tel que $X_0 + pX_1$ soit une racine de $F(X) = 0$ modulo p^2. Dans la formule de Taylor : $F(X_0 + pX_1) = F(X_0) + pF'(X_0)X_1 + \ldots$, nous pouvons clairement ignorer les termes en $p^k, k > 1$ (les pointillés) puisqu'ils s'annulent modulo p^2. Ensuite, $F(X_0) = 0 \bmod p$ implique que $F(X_0)$ (considéré modulo p^2) est un multiple de p. Nous calculons le vecteur $\lambda = F(X_0)/p$. Alors $F(X_0 + pX_1) = 0 \Rightarrow F(X_0) + pF'(X_0)X_1 = 0 \ [\bmod p^2] \Rightarrow \lambda + F'(X_0)X_1 = 0 \ [\bmod p]$, ce qui est un système linéaire, soluble modulo p. En fait, la remontée de Hensel n'est rien d'autre que la méthode de Newton, mais la remontée de Hensel donne le résultat exact (en supposant que X_0 soit exacte modulo p) contrairement à la méthode de Newton, qui ne fait que s'approcher de, et converger vers, la solution exacte.

Cette méthode peut être itérée pour calculer le développement P-adique de la racine, *i.e.,* les racines modulo $p^2, p^4, p^8, \ldots$, à partir de la racine X_0 modulo p.

Récapitulons : si $F(x) = 0$ est un système re-paramétré, alors chaque remontée de Hensel peut en tirer avantage. Notons toutefois que la re-paramétrisation n'est d'aucune aide pour calculer la racine initiale X_0 modulo p.

8. Re-paramétrisation et solveurs par intervalles

Les solveurs par intervalels calculent toutes les racines réelles d'un système $f(x) = 0$ à l'intérieur d'une boite initiale donnée. Ici f est un grand système re-paramétré.

Typiquement, ces solveurs gèrent une pile de boites à étudier. Cette pile est initialisée avec la boite initiale. Tant que cette pile n'est pas vide, la boite en sommet de pile, B, est dépilée, et étudiée. Cette étude essaie de réduire la boite, sans perdre aucune des racines contenues dans la boite.

- soit cette étude prouve que B ne contient aucune racine (par exemple B a été réduite à la boite vide) ;

- soit l'étude prouve que B contient une seule racine, régulière (le jacobien ne s'annule pas), par exemple en se fondant sur un test utilisant le théorème de Kantorovith. Alors cette racine est précisée par quelques itérations de la méthode standard (sans intervalle) de Newton-Raphson. Cette solution est insérée dans une liste de solutions ;

- soit la boite B après réduction est trop petite pour être subdivisée, et il est impossible de prouver qu'elle ne contient pas de solution ; peut-être contient elle une solution multiple ; ou peut-être contient elle plusieurs solutions très proches ; ou peut-être ne contient elle pas de solution du tout, mais la précision de l'ordinateur est insuffisante ; dans tous ces cas, la boite B est insérée dans une liste de boites résiduelles.

- soit, enfin, la boite B est encore volumineuse après réduction ; peut-être contient elle plusieurs racines réelles ? Alors B est coupée en deux (par exemple selon son côté le plus long), et les deux moities sont empilées.

Plusieurs méthodes ont été proposées dans la communauté de l'analyse par intervalles pour étudier la boite B, *i.e.,* trouver les racines contenues dans B ou réduire B (éventuellement à la boite vide) en préservant ses racines.

La méthode la plus directe résout par intervalles le système linéaire $f'(B)(x - x_0) = 0$, où x_0 est le centre de la boite étudiée B. D'abord $f'(B)$ est calculé, par intervalles, en exploitant son caractère épars. Puis nous calculons $x :=$ solve$(f'(B)x = f'(B)x_0$. Une difficulté surgit quand une des sous matrices de la diagonale de $f'(B)$ contient une matrice non inversible.

Une solution est le recours à l'évaluation centrée. Géométriquement, les points satisfaisant la i ième équation $f_i(x) =$

0 se trouvent sur une hypersurface ; dans la boite B, il est possible de calculer un hyperplan épais qui contienne cette hypersurface. Fig.7 montre un exemple 2D avec deux équations et deux inconnues $f_1(x_1,x_2) = f_2(x_1,x_2) = 0$. Chaque courbe est couverte par une trait droit et épais. Résoudre le système linéaire $a_{1,1}x_1 + a_{1,2}x_2 = [b_1, b_1']$, $a_{2,1}x_1 + a_{2,2}x_2 = [b_2, b_2']$ donne la boite réduite en pointillé (ce système n'a pas de structure R car il est trop petit). La matrice sous jacente ne contient pas d'intervalles, et est inversible avec probabilité 1, pour un système bien contraint. Seul le vecteur à droite contient des intervalles. Nous calculons les équations des hyperplans comme suit. De nouveau, nous partons de $f'(B)(x - x_0) = 0$. $f'(B)$ est calculé par intervalles, en exploitant son caractère creux. Soit J_0 le centre de la matrice par intervalles $f'(B)$, et soit $\Delta J := f'(B) - J_0$ une matrice intervalle, centrée. Alors :

$$f'(B)(x - x_0) = 0 \Rightarrow \quad (19)$$
$$(J_0 + \Delta J)(x - x_0) = 0 \Rightarrow \quad (20)$$
$$J_0(x - x_0) + \Delta J(x - x_0) = 0 \Rightarrow \quad (21)$$
$$J_0(x - x_0) + \Delta J(B - x_0) = 0 \Rightarrow \quad (22)$$
$$J_0\Delta x = -\Delta J(B - x_0) = V \Rightarrow \quad (23)$$
$$\Delta x = \text{solve}(J_0\Delta x = V), \quad (24)$$
$$x = x_0 + \Delta x \quad (25)$$

Puis nous calculons le vecteur intervalle $V := -\Delta J(B - x_0)$ en exploitant le caractère creux de ΔJ. Ensuite nous résolvons le système linéaire : $J_0\Delta x = V$. Remarquons que tous les éléments de la matrice J_0 sont des nombres flottants, non des intervalles. De plus J_0 a la structure R de f', donc cette résolution peut tirer avantage de la re-paramétrisation. Seul le vecteur V a des éléments qui sont des intervalles.

Enfin nous calculons $x := (x_0 + \Delta x) \cap B$. Si x est l'intervalle vide, alors B ne contient aucune racine.

Nous remarquons que l'idée de Gauss-Seidel peut être utilisée pour accélérer le calcul de $x := (x_0 + \Delta x) \cap B$: chaque élément x_k de x peut être réduit par $x_k := (x_0 + \Delta x)_k \cap B_k$, dès qu'il a été calculé. S'il est vide, alors la boite étudiée B est elle aussi réduite à la boite vide et ne contient donc aucune racine.

Une autre optimisation classique est le préconditionnement. Il limite l'effet enveloppant [Neu93] des calculs pr intervalles. Le système préconditionné est $g(x) = f'(x_0)^{-1}f(x) = 0$, et il est résolu à la place de $f(x) = 0$. f et g ont les mêmes racines. Par construction, $g'(x) = f'(x_0)^{-1}f'(x)$, donc $g'(x_0)$ est la matrice identité ; si f et g étaient linéaires, l'hyperplan numéro h d'équation $g_h(x) = 0$ serait perpendiculaire à l'axe x_h ; plus la boite est petite, et plus l'hypersurface $g_h(x) = 0$ est proche d'un hyperplan. Est il possible d'exploiter à la fois la re-paramétrisation et le préconditionnement ? La difficulté est que g' n'a plus la même structure R que f'.

Pour récapituler, les solveurs par intervalles peuvent eux aussi bénéficier de la re-paramétrisation, au moins en principe ; il faut cependant étudier l'effet enveloppant sur les systèmes re-paramétrés.

9. La re-paraméterisation à un plus haut niveau

Exploiter la re-paramétrisation au niveau des procédures omniprésentes d'algèbre linéaires a l'avantage de la simplicité, et de la factorisation de la programmation. Cela permet de réutiliser des solveurs pré-existants (Newton-Raphson, homotopie, solveurs par intervalles), au prix de quelques modifications mineures (à la limite, juste une recompilation), et de les accélérer d'un ordre de grandeur. Mais la re-paraméterisation peut aussi être utilisé à un niveau plus élevé. Par exemple, pour un solveur par intervalles, il semble possible de ne gérer que des boites selon U, les inconnues clefs ; X est ensuite calculé, par intervalles, en fonction de U. Toutefois, l'algorithme et le programme du solveur par intervalles doit être modifié de fond en comble.

10. Travaux futurs et questions émergentes

Plusieurs questions surgissent.

Les solveurs par intervalles peuvent ils bénéficier en même temps de la re-paramétrisation et du préconditionnement, ou de toute autre méthode limitant l'effet enveloppant des calculs par intervalles ?

Les solveurs fondés sur les bases tensorielles de Bernstein [FM12], et les solveurs utilisant les polytopes de Bernstein [FMF09], peuvent ils bénéficier, eux aussi, de la re-paramétrisation ?

Est il possible de détecter en temps polynômial de petits ensembles d'inconnues clefs ? Est il possible de re-écrire les systèmes peu denses d'équations, en y introduisant un petit ensemble d'inconnues clefs ?

La re-paramétrisation peut elle être utilisée dans d'autres secteurs, par exemple en algèbre linéaire pour le calcul des décompositions SVD, QR, etc, ou pour la résolution des problèmes de programmation linéaire, ou en calcul formel ?

Enfin, dans cet article, nous n'avons considéré que des systèmes d'équations, et pas des systèmes de contraintes géométriques. Explicitons la différence entre les deux : un système d'équations est structurellemnt bien contraint s'il a autant d'équations que d'inconnues, et, plus précisément, si son graphe biparti a un couplage parfait, ou davantage. Un système bien contraint (modulo les déplacements) de contraintes géométriques en 2D (respectivement en 3D) a trois (respectivement six) contraintes de moins que d'équations. Remarquons aussi que les contraintes géométriques (distances entre deux points, coplanarités de quatre points 3D) sont invariantes par changement de repère. Dans cet article, nous avons systématiquement converti les systèmes de

contraintes en systèmes d'équations, en fixant trois coordonnées en 2D et six coordonnées en 3D. Cette différence, apparemment mineure, entre systèmes d'équations et systèmes de contraintes, introduit de très nombreuses difficultés et complications. Par exemple, en mathématiques, la caractérisation combinatoire de la rigidité en 3D pour des systèmes de contraintes de distance (en gros, un système est rigide ssi il est bien contraint modulo les déplacements) est toujours inconnue. En informatique, cette différence complique la réduction des systèmes de contraintes géométriques, qui est discutée dans de très nombreux articles [JTNM06]. On peut penser que l'utilisation de la re-paramétrisation pour les contraintes est plus difficile que pour les équations, et que beaucoup d'articles vont y être consacrés.

11. Conclusion

Les méthodes dans [AAJM93] de décomposition des systèmes bien contraints d'équations ne s'appliquent pas aux systèmes re-paramétrés. De plus, quand il y a plus d'une inconnue clef, il est aujourd'hui difficile d'exploiter la re-paramétrisation [GHY04, GHY02, FS08, IMS11, MSI12].

Notre article généralise les méthodes de [AAJM93], si bien qu'elles s'appliquent aussi aux systèmes re-paramétrés. Nous proposons d'exploiter la re-paramétrisation au niveau le plus bas, celui des procédures omniprésentes d'algèbre linéaire. Ainsi de très nombreux solveurs peuvent, au prix de modification mineures, bénéficier de la re-paramétrisation. Les gains en termes de complexité algorithmique peuvent être spectaculaires : le coût d'une itération de Newton chute de $O(n^3)$ à $O(kn^2)$, ou même moins. Notre méthode permet donc d'utiliser la méthode de Newton-Raphson, et ses variantes (homotopie, solveur par intervalles), sur des systèmes bien plus grands.

Remerciements

This publication was made possible by NPRP grant #09-906-1-137 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

Cette publication a été rendue possible par le financement NPRP #09-906-1-137 du Qatar National Research Fund (un membre de la Qatar Foundation). Les déclarations faites ici n'engagent que la responsabilité de leurs auteurs.

Références

[AAJM93] AIT-AOUDIA S., JEGOU R., MICHELUCCI D. : Reduction of constraint systems. In *COMPUGRAPHICS* (Alvor, Algarve, Portugal, 1993), pp. 331–340.

[BH11] BETTIG B., HOFFMANN C. M. : Geometric constraint solving in parametric computer-aided design. *Journal of Computing and Information Science in Engineering. Vol. 11*, Num. 2 (2011), 021001.

[BR98] BRÜDERLIN B., ROLLER D. : *Geometric constraint solving and applications*. Springer, 1998.

[CLRS09] CORMEN T. H., LEISERSON C. E., RIVEST R. L., STEIN C. : *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.

[FM12] FOUFOU S., MICHELUCCI D. : Bernstein basis and its application in solving geometric constraint systems. *Reliab. Comput., this issue. Reliable Computing. Vol. 17* (2012).

[FMF09] FÜNFZIG C., MICHELUCCI D., FOUFOU S. : Nonlinear systems solver in floating-point arithmetic using lp reduction. In *Symposium on Solid and Physical Modeling* (2009), Bronsvoort W. F., Gonsor D., Regli W. C., Grandine T. A., Vandenbrande J. H., Gravesen J., Keyser J., (Eds.), ACM, pp. 123–134.

[FS08] FABRE A., SCHRECK P. : Combining symbolic and numerical solvers to simplify indecomposable systems solving. In *SAC* (2008), Wainwright R. L., Haddad H., (Eds.), ACM, pp. 1838–1842.

[GHY02] GAO X.-S., HOFFMANN C. M., YANG W.-Q. : Solving spatial basic geometric constraint configurations with locus intersection. In *Symposium on Solid Modeling and Applications* (2002), pp. 95–104.

[GHY04] GAO X.-S., HOFFMANN C. M., YANG W.-Q. : Solving spatial basic geometric constraint configurations with locus intersection. *Computer-Aided Design. Vol. 36*, Num. 2 (2004), 111–122.

[HJA05] HOFFMANN C. M., JOAN-ARINYO R. : A brief on constraint solving. *Computer-Aided Design and Applications. Vol. 2*, Num. 5 (2005), 655–663.

[Hof06] HOFFMANN C. M. : Summary of basic 2d constraint solving. *International Journal of Product Lifecycle Management. Vol. 1*, Num. 2 (2006), 143–149.

[IMS11] IMBACH R., MATHIS P., SCHRECK P. : Tracking method for reparametrized geometrical constraint systems. In *SYNASC* (2011), Wang D., Negru V., Ida T., Jebelean T., Petcu D., Watt S. M., Zaharie D., (Eds.), IEEE Computer Society, pp. 31–38.

[JTNM06] JERMANN C., TROMBETTONI G., NEVEU B., MATHIS P. : Decomposition of geometric constraint systems : a survey. *International Journal of Computational Geometry and Applications. Vol. 16*, Num. 5-6 (2006), 379–414.

[MSI12] MATHIS P., SCHRECK P., IMBACH R. : Decomposition of geometrical constraint systems with reparameterization. In *SAC* (2012), Ossowski S., Lecca P., (Eds.), ACM, pp. 102–108.

[Neu93] NEUMAIER A. : The wrapping effect, ellipsoid arithmetic, stability and confidence regions, 1993.