Revue Électronique Francophone d'Informatique Graphique, Volume 8, Numéro 2, pp. 79–91, 2014

Re-paramétrisation et réduction des systèmes irréductibles

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

[1] LE2I, Université de Bourgogne, Dijon, France
[2] Département d'informatique et ingénierie, CENG, Université du Qatar, Doha

Résumé

You have to solve a system of n non linear equations in n unknowns. The system is well constrained, and stucturally irreducible : it contains no strictly smaller well constrained subsystem. Its size is big, for example $n = 1000$, so the Newton-Raphson method is too slow. The most frustrating thing is that if you knew the values of a small number $k \ll n$ of some 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 devez résoudre un système de n équations non linéaires en n inconnues. Ce système est bien-contraint et structurellement irréductible : il ne contient pas de sous-système bien-contraint strictement plus petit. Il est de grande taille, par exemple $n = 1000$, si bien que la méthode de Newton-Raphson est trop lente. Le plus frustrant est que, si vous connaissiez la valeur de certaines inconnues clefs en très petit nombre (par exemple $k = 5 \ll n$), alors le système se décomposerait en de nombreux sous-systèmes bien-contraints 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, graphe biparti, Dulmage-Mendelsohn, König-Hall.

1. Introduction

La modélisation géométrique par contraintes [JMS07, 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, 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 décomposition, 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 niveau semble donc être le meilleur (du moins pour les plus paresseux d'entre nous) pour tirer avantage de la re-

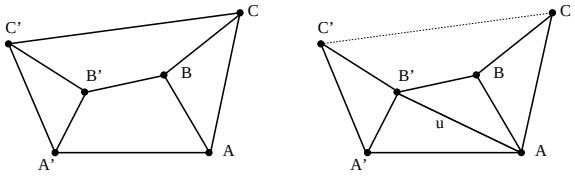

Figure 1: *A gauche : un système de contraintes géométriques en 2D ; chaque arête du graphe représente une contrainte de distance entre deux points, par exemple l'arête AB signifie que la distance AB est fixée. A droite : le système après re-paramétrisation : si $u = AB'$ était connu, alors le système serait réductible et facile à résoudre ; l'équation fixant la longueur CC' serait redondante et pourrait être ignorée.*

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.

La section 2 présente la re-paramétrisation sur des exemples. La section 3 présente la théorie du couplage dans les graphes bipartis : les méthodes de décomposition s'appuient sur cette théorie. La section 4 montre comment la décomposition en sous-systèmes accélère les procédures d'algèbre linéaire. La section 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 ; elle présente aussi la complexité algorithmique de cette méthode. La section 6 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 potentiellement le calcul symbolique, par exemple le calcul des bases de Gröbner. La section 7 montre que les solveurs par intervalles devraient aussi pouvoir tirer parti de la re-paramétrisation ; cependant l'effet enveloppant nécessite d'autres recherches. La section 8 envisage brièvement l'usage de la re-paramétrisation à un niveau supérieur. La section 9 présente les travaux futurs et les questions soulevées. La section 10 conclut.

2. La re-paramétrisation

2.1. Un exemple 2D simple

La figure 1 montre un système de contraintes géométriques en 2D. Les longueurs des arêtes sont fixées. 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 suffisamment génériques des longueurs — par exemple non nulles. Malheureusement, ce système est structurellement 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 deux équations à deux 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 fixant la longueur CC') est satisfaite. Quelques itérations de Newton, ou de la dichotomie, précisent la valeur solution de u. 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 d'abord au fait que $A(u), B(u), C(u)$ sont des multi-fonctions : algébriquement, des racines carrées apparaissent ; géométriquement, deux cercles peuvent avoir deux points d'intersection ; et ensuite 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, et géométriquement, pour l'intersection de deux cercles disjoints. Dernière difficulté, comment choisir l'intervalle des valeurs pour u ? Pour cet exemple, l'inégalité triangulaire fournit des intervalles de valeurs possibles pour u. 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, le problème n'admet 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 : la jacobienne est

de plein rang presque partout, donc le grand système a un nombre fini de solutions réelles, au moins pour les valeurs génériques des longueurs.

La valeur de u est inconnue, et le grand système est irréductible. En effet ses équations sont :

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

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

u	x_A	y_A	x_B	y_B	x_C	y_C
X	X	X	0	0	0	0
0	X	X	0	0	0	0
0	0	0	X	X	0	0
0	X	X	X	X	0	0
0	X	X	0	0	X	X
0	0	0	X	X	X	X
0	0	0	0	0	X	X

(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 deux 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.

Remarque : en fait, l'introduction de l'inconnue clef u, et de l'équation qui la définit, sont inutiles. Il suffit d'utiliser x_A comme inconnue clef. Après suppression de la première ligne (l'équation définissant u), et de la première colonne, la jacobienne devient :

x_A	y_A	x_B	y_B	x_C	y_C
X	X	0	0	0	0
0	0	X	X	0	0
X	X	X	X	0	0
X	X	0	0	X	X
0	0	X	X	X	X
0	0	0	0	X	X

(9)

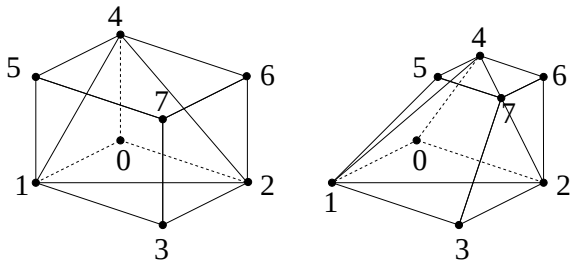

Figure 2: *Deux hexaèdres. Les longueurs des 12 arêtes (en gras) sont données. Les longueurs des trois diagonales (le triangle 124) sont les inconnues clefs. Si leurs valeurs étaient connues, le système serait réductible et facilement résolu, et les trois contraintes de longueur des arêtes incidentes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

et a toujours une structure triangulaire inférieure par blocs bien visible.

Par symétrie, n'importe laquelle des coordonnées $x_A, y_A, x_B, y_B, x_C, y_C$ peut être considérée comme l'inconnue clef. Il faut cependant permuter les lignes ou les colonnes de la jacobienne pour rendre visible sa structure triangulaire inférieure par blocs.

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), figure 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 est une inéquation ; elle élimine un continuum de solutions dégénérées, de dimension topologique 1, autrement dit une "courbe" d'hexaèdres aplatis [Imb13]. Pour simplifier, nous ignorons cette inéquation dans la suite.

Ce système a un nombre fini de solutions réelles, modulo les déplacements et les symétries. 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 voyons que $z_3 = 0$ à cause de la coplanarité des sommets 0, 1, 2, 4, et que $x_1 = l_{01}$ où l_{01} est la longueur fixée pour l'arête 01 ; ceci satisfait deux contraintes que nous pouvons donc éliminer. Il nous reste à résoudre un système de 16 équations et 16 inconnues. Ce dernier est irréductible, concluent 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 des coordonnées pour les sommets du triangle 012

	l_{12}	l_{24}	l_{41}	x_1	x_2	y_2	x_3	y_3	z_3	x_4	y_4	z_4	x_5	y_5	z_5	x_6	y_6	z_6	x_7	y_7	z_7
$D(0,1)$				X																	
$D(0,2)$					X	X															
$D(1,2)$	X			X	X	X															
$D(1,3)$				X			X	X	X												
$D(2,3)$					X	X	X	X	X												
$C(0,1,2,3)$				X	X	X	X	X	X												
$D(0,4)$										X	X	X									
$D(1,4)$			X	X						X	X	X									
$D(2,4)$		X			X	X				X	X	X									
$D(1,5)$				X									X	X	X						
$D(4,5)$										X	X	X	X	X	X						
$C(0,1,4,5)$				X						X	X	X	X	X	X						
$D(2,6)$					X	X										X	X	X			
$D(4,6)$										X	X	X				X	X	X			
$C(0,2,4,6)$					X	X				X	X	X				X	X	X			
$C(1,3,5,7)$				X			X	X	X				X	X	X				X	X	X
$C(2,3,6,7)$					X	X	X	X	X							X	X	X	X	X	X
$C(4,5,6,7)$										X	X	X	X	X	X	X	X	X	X	X	X
$D(3,7)$							X	X	X										X	X	X
$D(5,7)$													X	X	X				X	X	X
$D(6,7)$																X	X	X	X	X	X

Table 1: *La structure du jacobien du grand système re-paramétré, dans l'exemple de l'hexaèdre. Les cases vides sont nulles. Les trois premières colonnes de la matrice jacobienne concernent les trois inconnues clefs. Les trois dernières lignes concernent les équations oubliées. Comme d'habitude, on a imposé la valeur de six inconnues : $x_0 = y_0 = z_0 = y_1 = z_1 = z_2 = 0$. $D(a,b)$ est la contrainte de distance entre les points A et B. $C(a,b,c,d)$ désigne la contrainte de coplanarité des points a,b,c,d. La structure triangulaire inférieure par blocs est bien visible, grâce à la permutation choisie pour les lignes (les équations) et les colonnes (les inconnues).*

sont facilement trouvées ; le sommet 4 est l'intersection de trois sphères de centres et de rayons connus ; ensuite 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 apparaît 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 implicite : $F(U,X) = 0,\ G(X) = 0$, ou même $F(U,X) = 0,\ G(U,X) = 0$, qui est la plus générale. C'est cette dernière qui est utilisée dans la suite de cet article.

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. Dans leurs articles fondateurs, Gao, Hoffmann et Yang [GHY04] traitent seulement le cas d'une seule inconnue clef, et montrent que de nombreux problèmes 3D deviennent solubles par cette re-paramétrisation.

Notre article présente une méthode simple, utilisable pour un nombre quelconque d'inconnues clefs ; la seule restric-

tion est que le nombre d'inconnues clefs, k, doit rester petit devant n, la taille du système initial.

3. La théorie du couplage

La décomposition des systèmes d'équations est fondée sur la théorie du couplage. Le livre de Lovasz et Plummer : "Matching theory" détaille cette théorie ; le lecteur trouvera dans ce livre les liens entre couplage et matroïde, les théorèmes de König-Hall, la décomposition de Dulmage et Mendelsohn, etc. Cette section présente l'essentiel de 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 si et seulement si 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 de taille maximale. 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. La taille d'un couplage et son nombre d'arêtes ; un couplage est de taille maximale, s'il n'existe pas de couplage de taille plus grande. Un couplage est *maximal* s'il est maximal pour l'inclusion : il n'est strictement inclus dans aucun autre couplage. Un couplage maximal n'est pas toujours de taille maximale, mais tout couplage de taille maximale est bien sûr maximal. Un couplage est *parfait* si tous les sommets sont saturés. Un couplage parfait est maximal et de taille maximale.

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 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

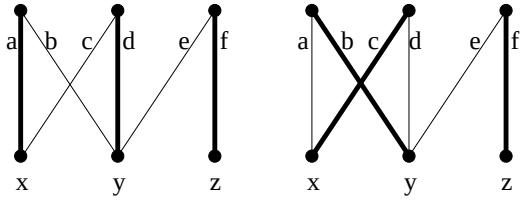

Figure 3: *Le graphe biparti d'un système linéaire $ax+by=r_1, cx+dy=r_2, ey+fz=r_3$, et ses deux couplages parfaits adf et bcf. Le déterminant de la matrice sous-jacente est $adf-bcf$.*

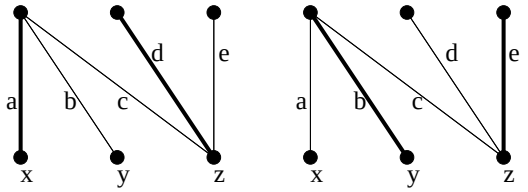

Figure 4: *Le système $F_1(x,y,z)=F_2(z)=F_3(z)=0$ est structurellement mal-contraint. "Structurellement" signifie que le système est mal contraint quelles que soient les valeurs de a,b,c,d,e. Aussi son graphe biparti n'a-t-il aucun couplage parfait. Les couplages de taille maximale contiennent deux 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).*

être plus grand que le nombre d'arêtes d'un couplage de taille maximale du graphe biparti associé. En général, le rang de la matrice est égal au cardinal de n'importe lequel des couplages de taille maximale du graphe biparti associé. Dans les cas particuliers, ou dégénérés, le rang de la matrice (ou du système) est inférieur à la cardinalité des couplages de taille maximale. La bijection entre chaque couplage parfait et chaque terme du déterminant d'une matrice (par exemple jacobienne) est illustrée en figure 3 pour la matrice :

$$M=\begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & e & f \end{pmatrix} \tag{10}$$

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 bijection entre couplage parfait et termes du déterminant vaut pour toute matrice carrée. En effet, soit M une matrice carrée, et soit G le graphe biparti associé ; autrement dit, G est le graphe biparti associé au système linéaire $Mx=b$, où b est un vecteur donné générique (non nul). Une arête lie le sommet de l'équation de la ligne l à un sommet de l'inconnue c *ssi* $M_{l,c}$ est non nul. Alors le déterminant de M

est $\sum_\sigma \text{pair}(\sigma) \prod_{l=1}^{n} M_{l,\sigma(l)}$ où σ parcourt toutes les permutations de $1,\ldots n$, et où $\text{pair}(\sigma)$ vaut $+1$ ou -1 selon que la permutation σ est paire ou impaire. Rappelons qu'une permutation σ est paire (respectivement, impaire) *ssi* σ est la composition d'un nombre pair (respectivement, impair) d'échanges de deux éléments distincts. Tout terme $\prod_{l=1}^{n} M_{l,\sigma(l)}$ non nul ne contient que des $M_{l,\sigma(l)}$ non nuls, donc la permutation σ donne un couplage parfait dans G : ce couplage associe au sommet d'une équation l le sommet d'une inconnue $\sigma(l)$.

Cette bijection entre couplage parfait et termes du déterminant explique pourquoi, lorsqu'il n'y a aucun couplage parfait, le déterminant est identiquement nul, comme pour le système structurellement mal-contraint de la figure 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 de taille maximale (parfait pour les 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 de taille maximale. Ceci est illustré sur la figure 5 pour l'exemple de la figure 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 de taille maximale 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.

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 *DAG* (*Directed Acyclic Graph*, graphe sans cycle) de sous-systèmes, dont les arcs indiquent les dépendances entre sous-systèmes. Ces *DAG* sont les graphes réduits en figure 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.

Pour conclure cette section, mentionnons une petite différence entre le graphe biparti d'un système non linéaire $F(X)=0$, et celui de sa jacobienne $J=F'(X)$, autrement dit du système linéaire $J\Delta x = -F(X)$ (où X est donné, $J=F'(X)$ est calculé, et ΔX est inconnu) qui est résolu par la méthode de Newton-Raphson. Le graphe biparti de J peut avoir quelques arêtes de moins que le graphe biparti

†. Pour la distinction entre temps fortement et faiblement polynomial, voir par exemple [Sch03].

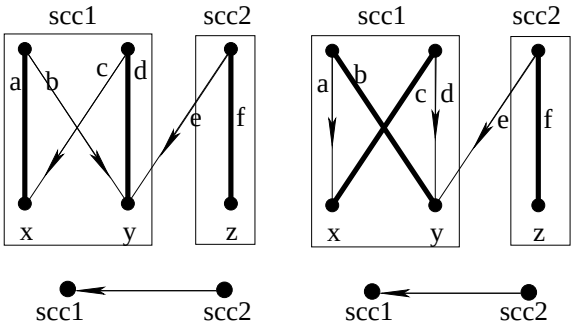

Figure 5: *Au dessus : les deux couplages parfaits du système de la figure 3, et les composantes fortement connexes induites dans le graphe orienté ; les arêtes du couplage sont orientées dans les deux sens, et les autres de haut en bas (des équations vers les inconnues). En dessous : dans le graphe 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.*

du système $F(X)=0$. En effet, si la l-ième équation F_l de F utilise la c-ième inconnue x_c uniquement dans des monômes de degré au plus un en x_c, alors la dérivée $\partial F/\partial x_c$ de F_l relativement à x_c est nul, donc $J_{l,c}$ est nul, et il n'y a pas d'arête entre le sommet de la l-ième équation et le sommet de la c-ième inconnue dans le graphe biparti de la jacobienne. Les ensembles d'inconnues clefs pour le système linéaire $J\Delta x = -F(X)$ sont donc un peu moins contraints, aussi peuvent-ils parfois être plus petits que pour le système non linéaire $F(X)=0$. Ceci est un autre argument pour l'utilisation de la re-paramétrisation au niveau de l'algèbre linéaire, qu'il convient de pondérer : dans les exemples présentant la re-paramétrisation, les graphes bipartis de F et de J sont égaux.

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 triangulaire inférieure par blocs rend visible la décomposition en sous-systèmes bien-contraints irréductibles. Cette décomposition 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 résoudre un système linéaire est en $O(n^2)$ pour une matrice triangulaire inférieure, et en $O(n^3)$ dans le cas général en utilisant la méthode des pivots de Gauss [CLRS09].

Cette section suppose que la matrice M est triangulaire

inférieure par bloc, comme ceci par exemple :

$$M = \begin{pmatrix} M_{1,1} & 0 & 0 \\ M_{2,1} & M_{2,2} & 0 \\ M_{3,1} & M_{3,2} & M_{3,3} \end{pmatrix} \quad (11)$$

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 en section 5), ou des intervalles (quand utilisés par des solveurs par intervalles en section 7), ou des éléments d'un corps fini ou d'un corps p-adique (sectiion 6).

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.

$$M^{-1} = K = \begin{pmatrix} K_{1,1} & 0 & 0 \\ K_{2,1} & K_{2,2} & 0 \\ K_{3,1} & K_{3,2} & K_{3,3} \end{pmatrix} \quad (12)$$

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

$$K_{l,c} = 0 \text{ quand } c > l \quad (13)$$

$$K_{l,l} = M_{l,l}^{-1} \quad (14)$$

$$K_{l,c} = -K_{l,l} \sum_{i=c}^{l-1} M_{l,i} K_{i,c} \text{ quand } l > c \quad (15)$$

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

$$\begin{aligned} X_1 &:= \text{solve}(M_{1,1}X_1 = B_1), \text{ puis} \\ X_2 &:= \text{solve}(M_{2,2}X_2 = B_2 - M_{2,1}X_1), \text{ puis} \\ X_3 &:= \text{solve}(M_{3,3}X_3 = B_3 - (M_{3,1}X_1 + M_{3,2}X_2)), \textit{ i.e.,} \\ X_l &:= \text{solve}(M_{l,l}X_l = B_l - \sum_{i=1}^{l-1} M_{l,i}X_i) \end{aligned}$$

Plus les blocs diagonaux sont petits et donc nombreux, plus le nombre de blocs nuls sous la diagonale est important, et plus grande est l'accélération. Pour évaluer cette accélération, étudions la complexité dans un cas simplifié : M est de taille n par n et tous les blocs diagonaux sont de tailles égales, t lignes par t colonnes ; il y a donc $b = n/t$ blocs diagonaux ; soit β le nombre des matrice blocs sous la diagonale et non nulles ; donc $0 \leq \beta \leq b(b-1)/2 \in O(b^2) = O(n^2/t^2)$. Alors, la résolution de $MX = b$ nécessite β produits d'une matrice (t,t) par un vecteur colonne de taille t, ce qui coûte βt^2, et autant d'additions de vecteurs colonnes de taille t, ce qui est de coût moindre : βt, et enfin b résolutions de systèmes linéaires $X_l = B_l - \sum_{i=1}^{l-1} M_{l,i}X_i), l = 1, \ldots b$, ce qui coûte $O(bt^3)$, à raison de $O(t^3)$ par inversion (les inversions à la Strassen ne sont pas pertinentes pour de petites matrices t par t ; de plus, nous allons supposer $t \in O(1)$). D'où un coût total de $O(\beta t^2 + bt^3) = O((\beta + bt)t^2)$. Or $bt = n$. Donc le coût est $O((\beta + n)t^2)$.

Supposons maintenant t assez petit (par exemple $t \leq 10$) pour qu'il puisse être considéré comme une constante : $t \in O(1)$. Alors la complexité totale de la résolution de $MX = B$ est en $O(\beta + n)$. Rappelons que $0 \leq \beta \leq b(b-1)/2 = O((n/t)^2) = O(n^2)$. Ainsi, même quand il n'y a aucun bloc nul sous la diagonale, la complexité tombe de n^3 à n^2. Si β est de l'ordre de n, la complexité chute de n^3 à n ; ceci peut arriver avec les systèmes de contraintes géométriques, car chaque contrainte, par exemple celle de la distance entre deux points, dépend d'un nombre constant de variables.

L'accélération due à la décomposition peut donc être considérable.

Dans la suite nous utiliserons $\alpha = \beta + n$: α est donc le nombre de matrices blocs non nulles ; nous dirons que résoudre est en $O(\alpha)$: c'est vrai car nous supposons que t est une constante.

Le coût de la résolution optimale d'un système linéaire creux, et la stratégie optimale de résolution (par exemple comment choisir les pivots de Gauss pour résoudre le système le plus vite possible ?) sont étudiés plus en détail dans la thèse de M. Bomhoff [Bom13]. Ces problèmes, et les méthodes combinatoires utilisées, ne considèrent que le graphe biparti du système linéaire.

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

Cette section montre que la re-paramétrisation accélère les calculs d'algèbre linéaire effectués par la méthode de Newton-Raphson, ou une de ses variantes, comme la méthode de Newton-Raphson amortie (*damped Newton*) ou l'homotopie. Dans la suite, nous ne considérons que la méthode de Newton-Raphson.

Nous supposons que le système d'équations est bien-contraint ; l'ensemble des inconnues clefs est connu ; la structure de la matrice jacobienne du système a été calculée par les méthodes de [AAJM93] ; elle restera inchangée lors des itérations de la méthode de Newton-Raphson.

Chaque itération de Newton-Raphson résout un système linéaire où la matrice a la structure (déjà vue en (Eq.8) ou dans l'exemple de la figure 1) montrée en figure 6.

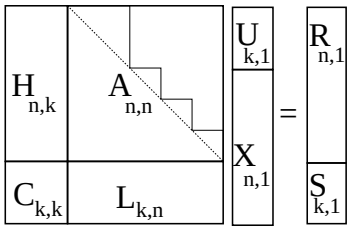

Figure 6: *La structure de la jacobienne du grand système re-paramétré.*

En formules :

$$HU + AX = R, \quad CU + LX = S \tag{16}$$

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.

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 si et seulement si les tailles de leurs blocs H, A, C, L sont égales et les structures de bloc de leur sous-matrice A sont égales.

On déduit de (Eq.16) :

$$(C - LA^{-1}H)U = S - LA^{-1}R \tag{17}$$

Preuve :

$$\begin{aligned} & (\text{Eq.16}) \Rightarrow CU + LX = S, \quad X = A^{-1}(R - HU) \\ \Rightarrow \quad & CU + LA^{-1}(R - HU) = S \\ \Rightarrow \quad & (C - LA^{-1}H)U = S - LA^{-1}R \end{aligned}$$

Nous allons résoudre cette dernière équation linéaire en U, puis en déduire X, ceci en évitant l'inversion de A. On suppose A de taille $n \times n$. 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 $Ax = B$, comme vue à la section précédente. Résoudre $Ax = B$ (pour B donné) coûte $O(\alpha)$, le nombre de blocs non nuls dans la matrice A. Les équations (Eq.16) (Eq.17) sont résolus en U et X comme suit.

1 Résoudre : $Z_{n\times 1} := \text{solve}(A_{n\times n} Z_{n\times 1} = R_{n\times 1})$ coûte α. Ceci donne $Z = A^{-1}R$, qui apparaît dans (Eq.17), en nous évitant l'inversion de la matrice A.
2 Résoudre : $K_{n\times k} := \text{solve}(A_{n\times n} K_{n\times k} = H_{n\times k})$. Ceci nécessite la résolution de k systèmes linéaires, un pour chaque colonne $c = 1, \ldots k$ de K, et la colonne correspondante de H. Ceci coûte donc $O(k\alpha)$. Ceci donne $K = A^{-1}H$, qui apparaît dans (Eq.17), en nous évitant l'inversion de la matrice A.
3 Calculer $C - LA^{-1}H = C_{k\times k} - L_{k\times n} K_{n\times k}$ coûte $O(k^2 n)$. Ce terme intervient dans l'étape 5.
4 Calculer $S - LA^{-1}R = S_{k\times 1} - L_{k\times n} Z_{n\times 1}$ coûte $O(kn)$. Ce terme intervient dans le membre droit de (Eq.17) et à l'étape 5.
5 Résoudre : $U_{k\times 1} := \text{solve}((C - LA^{-1}H)_{k\times k} U_{k\times 1} = (S - LA^{-1}R)_{k\times 1})$, où seul U est inconnu, coûte $O(k^3)$. C'est l'équation (17). Il est inutile d'optimiser cette étape car k est petit. Ceci nous donne U.
6 Calculer $R_{n\times 1} - H_{n\times k} U_{k\times 1}$ coûte $O(nk)$. Ce terme intervient dans l'étape 7.
7 Résoudre $X_{n\times 1} := \text{solve}(A_{n\times n} X_{n\times 1} = (R - HU)_{n\times 1})$, où seul X est inconnu, coûte $O(\alpha)$, en évitant l'inversion de A. C'est la première équation du système (16). Ceci nous donne X.

U et X sont maintenant connus. Le coût total de la résolution d'un système linéaire re-paramétré est donc : $O(k(\alpha + kn + k^2))$. Comme $k \ll n$, et comme $n \leq \alpha \leq n(n-1)/2 \in O(n^2)$, ce coût est donc compris entre $k^2(n+k)$ dans le meilleur des cas et $O(kn^2 + k^n + k^3) = O(kn^2)$ dans le pire des cas. Ce coût est toujours inférieur à n^3, le coût de la résolution d'un système linéaire par les méthodes standard (pivots de Gauss, décomposition LU) ; il est même inférieur au coût de l'inversion par la méthode de Strassen : $O(n^{\log_2 7}) \approx O(n^{2.8\ldots})$ et à celui de l'inversion par la méthode de Coppersmith–Winograd : $O(n^{2.375\ldots})$ (voir [CLRS09] pour les détails. Il y est montré que l'inversion d'une matrice $n \times n$ a asymptotiquement le même coût qu'un produit de deux matrices $n \times n$).

Une remarque en passant : nous supposons pour simplifier que $k \ll n$, et nous n'abordons pas la question de la taille maximale de k pour que l'accélération due à la re-paramétrisation reste intéressante. L'analyse en complexité précedente suggère une borne pour k : si $k = O(n^{0.375\ldots})$, alors notre méthode a la même complexité que celle de Coppersmith–Winograd : $O(n^{2.375\ldots})$. Cette borne pour k est intéressante car il est probablement plus facile de trouver des ensembles d'inconnues clefs de taille $O(n^{0.375\ldots})$ que de taille $O(1)$ ou $O(\log n)$.

6. Le cas p-adique

Les méthodes p-adiques sont utilisées en analyse p-adique, mais aussi pour résoudre des problèmes diophantiens [Coh06, ST88, BP92, Box14, SCL59] ou de calcul formel : calcul du PGCD ou de la factorisation de polynômes [Gal10, Coh93], ou calcul de bases de Gröbner [Fau99].

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 dans les p-adiques [Fau99] ‡.

Cette méthode peut être itérée pour calculer le développement p-adique de la racine, *i.e.,* les racines modulo $p, p^2, p^3, \ldots$ (ou $p^2, p^4, p^8, \ldots$) 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.

7. Re-paramétrisation et solveurs par intervalles

Cette section montre que les solveurs par intervalles (ALIAS [COP12], RealPaver [GB06], Ibex [Cha14] and QUIMPER [CJ11]) peuvent tirer parti de la re-paramétrisation. Les solveurs par intervalles utilisent des méthodes de l'analyse par intervalles [Mer00, Tro09, MKC09, JKDW01], qui calculent toutes les racines réelles d'un système $f(x) = 0$ à l'intérieur d'une boîte initiale donnée. Ici f est un grand système re-paramétré, bien-contraint.

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

- soit cette étude prouve que B ne contient aucune racine ; par exemple B a été réduite à la boîte 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 Kantorovich [CM12]. 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 boîte 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 boîte B est insérée dans une liste de boîtes résiduelles.

- soit, enfin, la boîte B est encore volumineuse après réduction ; peut-être contient-elle plusieurs racines réelles ? Alors elle est coupée en deux, par exemple selon son côté le plus long (d'autres choix ont été étudiés [COP12, Mer00, Tro09]), et les deux moitiés sont empilées.

Plusieurs méthodes ont été proposées dans la communauté de l'analyse par intervalles pour réduire la boîte B, éventuellement à la boîte vide, en préservant ses racines. Citons l'opérateur de Krawczyck-Moore, ou celui de Sengupta-Hansen ([Tro09], pg 25-26).

La méthode la plus directe résout par intervalles un système linéaire. Soit x_0 le centre de la boîte B. Alors :

$$f(x \in B) = f(x_0 + (x - x_0)) \ \in \ f(x_0) + f'(B)(x - x_0) \quad (18)$$

La figure 7 à gauche donne une interprétation géométrique de cette formule, pour un problème 1D. Nous cherchons $x \in B$ solution de $f(x_0) + f'(B)(x - x_0) = 0$. Posons $\Delta x = x - x_0$; Δx est la solution du système linéaire par intervalles : $f'(B)\Delta x = -f(x_0)$. $f'(B)$ est d'abord calculée, par intervalles, en profitant du caractère creux de la jacobienne f'. Puis nous calculons $\Delta x := \text{solve}(f'(B)\Delta x = -f(x_0))$ en exploitant la R structure de la jacobienne, avec les méthodes en section 5.

Une difficulté surgit quand une des sous-matrices de la diagonale de jacobienne $f'(B)$ contient une matrice non inversible. C'est le cas dans l'exemple 1D de la figure 7.

Une solution est le recours à la forme centrée. Soient $J = f'(B)$, et J_0 le centre de J qui est une matrice inversible avec probabilité 1. Il est aussi possible d'utiliser $J_0 = f'(x_0)$. Soit ensuite $\Delta J = J - J_0$. Alors :

$$\begin{aligned}
& f(x_0) + J\Delta x = 0 \\
\Rightarrow \ & f(x_0) + (J_0 + \Delta J)\Delta x = 0 \\
\Rightarrow \ & f(x_0) + J_0\Delta x + \Delta J\Delta x = 0 \\
\Rightarrow \ & f(x_0) + J_0\Delta x + \Delta J(x - x_0) = 0, \ x \in B \\
\Rightarrow \ & f(x_0) + J_0\Delta x + \Delta J(B - x_0) = 0 \\
\Rightarrow \ & J_0\Delta x = -f(x_0) - \Delta J(B - x_0) \qquad (19)
\end{aligned}$$

Cette dernière équation est un système linéaire, d'inconnue Δx ; les éléments de la matrice J_0 sont des nombres flottants, non des intervalles. Avec probabilité 1, J_0 est inversible. De plus J_0 a la structure R de f', donc la résolution du système linéaire peut bénéficier de la re-paramétrisation. Seul le vecteur du membre droit de l'équation $-(f(x_0) + \Delta J(B - x_0)$ a des coordonnées intervalles. Géométriquement, l'hypersurface de chaque équation est couverte par un hyperplan épais,

‡. "Note that the p-adic method is also an iterative algorithm (in fact this is a Newton algorithm)" [Fau99].

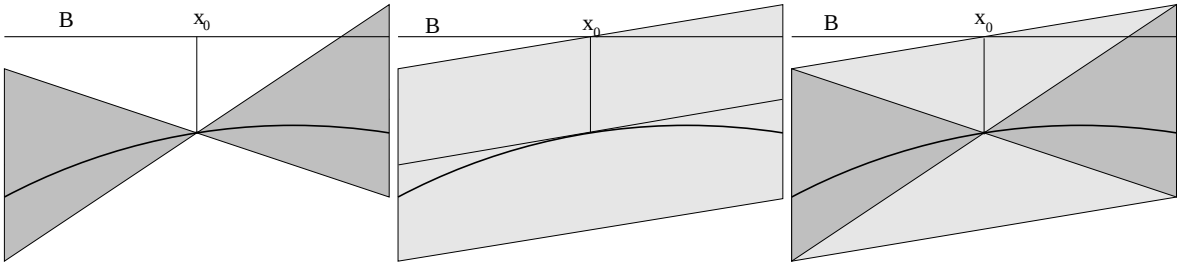

Figure 7: *Une fonction peut être encadrée de deux manières par une fonction linéaire par intervalles. A gauche, $f(x) \in f(x_0) + f'(B)(x - x_0) = -1 + [-1/3, 2/3]x$: c'est l'équation des deux triangles grisés. Au milieu, $f(x) \in x/6 - [0,2]$: c'est l'équation de la bande grisée. A droite, les deux figures sont superposées : la bande grisée est l'enveloppe convexe des deux triangles grisés.*

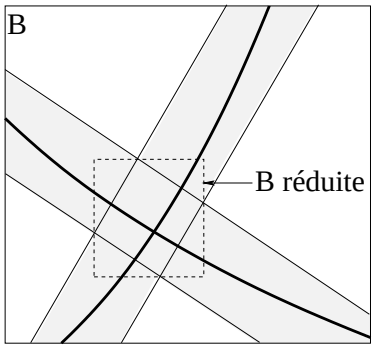

Figure 8: *La linéarisation par intervalles d'un système en 2D dans une boîte B. Les deux courbes sont couvertes par des droites épaisses, les bandes grisées. Résoudre donne la boîte réduite en pointillé.*

d'épaisseur constante ; c'est l'enveloppe convexe des cônes de la linéarisation par intervalles précédente.

Un exemple simple, en 1D, est illustré sur la figure 7. $B = [-2,2]$, $x_0 = 0$, $f(x_0) = -1$, et $f'(B) = [-1/3, 2/3]$. La première linéarisation par intervalles est : $f(x_0 + \Delta x) \in f(x_0) + f'(B)\Delta x$, ce qui donne $-1 + [-1/3, 2/3]\Delta x = 0$. Géométriquement, l'arc de la courbe est couvert par les deux triangles grisés, à gauche de la figure. $f'(B)$ n'est pas inversible. Le milieu de la figure illustre la seconde linéarisation par intervalles, la forme centrée ; l'arc de la courbe est couvert par une droite épaisse, la bande grisée, d'équation : $J_0\Delta x = -f(x_0) - \Delta J(B - x_0)$; numériquement : $1/6x - [0,2] = 0$. La figure à droite superpose les deux encadrements. On remarque que l'intersection de la droite Ox avec la droite épaisse est plus grande qu'avec les deux triangles grisés.

La figure 8 montre un exemple 2D.

Une fois calculé Δx, la boîte B est mise à jour, par $B := B \cap (x_0 + \Delta x)$. Quand cette intersection est vide, B ne contient aucune racine. Si la boîte n'est pas réduite, elle est coupée en deux, par exemple selon son côté le plus long. La bissection est inévitable car elle seule peut séparer les racines.

L'idée de Gauss-Seidel peut être utilisée pour accélérer le calcul de $B \cap (x_0 + \Delta x)$: chaque i-ième coordonnée B_i de B peut être réduite par $B_i := B_i \cap (x_0 + \Delta x)_i$, dès qu'elle a été calculée ; la suite des calculs utilise la dernière valeur, la plus précise, de B_i. Si un des B_i est vide, alors B est vide elle aussi, et ne contient donc aucune racine. Cette idée est utilisée dans l'opérateur de Hansen-Sengupta.

Une autre optimisation classique, utilisée elle aussi dans l'opérateur de Hansen-Sengupta, est le préconditionnement. Ce dernier limite l'effet enveloppant [Neu93] des calculs par 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 boîte 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' ; en effet, le produit de deux matrices ayant la même structure R n'a pas la structure R, en général.

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

8. La re-paramétrisation à un plus haut niveau

Exploiter la re-paramétrisation au niveau des procédures omniprésentes d'algèbre linéaire a l'avantage de la simplicité, et de la factorisation de la programmation. Cela permet de réutiliser des solveurs préexistants (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 ou deux. Mais la re-paramétrisation peut aussi être utilisée à un niveau plus élevé. Par exemple, pour un solveur par intervalles, il semble possible de ne gérer que des boîtes 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 doivent être grandement modifiés.

9. Résultats préliminaires, et questions émergentes

Pour résoudre le problème 3D du pentaèdre [BCG*14, BCMF14], nous avons utilisé une re-formulation du système de contraintes, qui réduit la taille du système à résoudre, de douze (12 équations et 12 inconnues) à trois (3 équations et 3 inconnues). Avec cette re-formulation, un solveur par intervalles (ALIAS) et une méthode de Newton-Raphson standard résolvent le problème plus de 40 fois plus rapidement. Cette re-formulation est une re-paramétrisation *ad hoc*, en ce sens qu'elle ne se généralise pas à d'autres problèmes géométriques. Nonobstant, cet exemple donne une indication sur l'accélération que pourra apporter la re-paramétrisation, quand les méthodes que nous avons proposées dans cet article seront programmées. Cette section mentionne maintenant plusieurs questions non traitées dans cet article.

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 ? Par exemple, parmi les multiples opérateurs de contraction en analyse par intervalles (opérateur de Krawczyck-Moore, opérateur de Hansen-Sengupta, etc) ou de propagation d'intervalles (*Box consistency* : 2B, Box, BC4, HC4, Mohc, etc [Tro09]), y en a-t-il un qui soit ou qui puisse être adapté pour tirer parti de la re-paramétrisation ? La même question se pose pour les solveurs fondés sur les bases tensorielles de Bernstein [FM12a], et les solveurs utilisant les polytopes de Bernstein [FMF09].

Quelles équations redondantes oublier ? Par exemple, on peut penser qu'il vaut mieux oublier les équations de plus haut degré.

Pour simplifier, nous avons supposé que le nombre d'inconnues clefs k était une constante, ou petit devant n le nombre d'inconnues et d'équations. Vraisemblablement, un k de l'ordre de $\log n$, ou $\sqrt{n}$ reste intéressant. Peut-on préciser la valeur maximale de k qui reste intéressante ?

Nous n'avons pas abordé le problème du calcul d'un petit ensemble d'inconnues clefs, que ce soit pour les systèmes d'équations, ou les systèmes de contraintes géométriques. Rappelons seulement que l'équipe de Strasbourg [FS08, Thi10, MSI12, Imb13] a proposé des algorithmes de re-paramétrisation des systèmes de contraintes géométriques en temps polynomial ; ces algorithmes fournissent de petits (peut-être non minimaux) ensembles d'inconnues clefs. La généralisation de ces méthodes à des systèmes d'équations peu denses plus généraux, autres que des contraintes géométriques, est un problème ouvert. Dans sa thèse, M. Bomhoff [Bom13] étudie quelques problèmes voisins concernant les systèmes d'équations ; typiquement, le calcul de la solution optimale de ces problèmes est NP-dur, mais il semble possible de calculer en temps polynomial des solutions proches de l'optimum.

La re-paramétrisation peut-elle être utilisée dans d'autres domaines, 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 ?

Par simplicité, nous avons utilisé des méthodes combinatoires pour décomposer les systèmes. Malheureusement, ces méthodes ne détectent que les dépendances structurelles, et pas les dépendances dues à des théorèmes géométriques. L'interrogation d'un témoin [MF09, FM12b, MST*10] permet de détecter toutes les dépendances et aussi de décomposer, sous réserve que soit connu un témoin typique de la solution souhaitée [KMF]. D'où la question : peut-on remplacer les méthodes combinatoires par la méthode du témoin pour tirer parti de la re-paramétrisation ?

Enfin, dans cet article, nous n'avons considéré que des systèmes d'équations, bien-contraints, et pas des systèmes de contraintes géométriques : nous avons systématiquement converti les systèmes de contraintes en systèmes d'équations, en fixant trois coordonnées en 2D et six coordonnées en 3D par des équations *ad hoc*. Ces dernières dépendent du repère et ne sont donc pas des contraintes géométriques. Cette différence, apparemment bénigne, entre équations et contraintes géométriques, introduit d'innombrables difficultés et complications.

Ainsi, en mathématiques, la caractérisation combinatoire des systèmes de contraintes géométriques bien-contraints en 3D (dits rigides, ou isostatiques) est toujours inconnue, même dans les cas les plus simples, où toutes les contraintes spécifient des distances génériques entre points. En 2D, la caractérisation combinatoire de la rigidité est donnée par le théorème de Laman, mais il est extrêmement restrictif ; par exemple, il suppose que les seules contraintes sont des distances entre points, et que ces distances sont algébriquement indépendantes, ce qui interdit les alignements, les cocyclicités, etc.

En informatique, cette différence complique fortement la décomposition par des méthodes combinatoires des systèmes de contraintes géométriques [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 être consacrés à ces deux questions.

10. 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, 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. Notre méthode permet donc d'utiliser la méthode de Newton-Raphson et ses variantes (homotopie, solveur par intervalles) sur des systèmes bien plus grands.

Acknowledgements

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.

Remerciements

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.

Nous remercions chaleureusement les rapporteurs pour leurs commentaires.

Références

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

[BCG*14] BARKI H., CANE J.-M., GARNIER L., MICHELUCCI D., FOUFOU S. : Solving the pentahedron problem. *Computer-Aided Design* (2014).

[BCMF14] BARKI H., CANE J., MICHELUCCI D., FOUFOU S. : New geometric constraint solving formulation : Application to the 3d pentahedron. In *Image and Signal Processing - 6th International Conference, ICISP 2014, Cherbourg, France, June 30 - July 2, 2014. Proceedings* (2014), Elmoataz A., Lezoray O., Nouboud F., Mammass D., (Eds.), vol. 8509 de *Lecture Notes in Computer Science*, Springer, pp. 594–601.

[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.

[Bom13] BOMHOFF M. J. : *Bipartite graphs and the decomposition of systems of equations*. PhD thesis, University of Twente, Enschede, January 2013.

[Box14] BOX J. : *An introduction to Skolem's p-adic method for solving Diophantine equations*. Bachelor thesis, dirigée par Dr. Sander Dahmen, Korteweg-de Vries Instituut voor Wiskunde Faculteit der Natuurwetenschappen, Wiskunde en Informatica, Universiteit van Amsterdam, 2014. Disponible sur internet.

[BP92] BAKER A. J., PLYMEN R. J. : *P-adic Methods and Their Applications*. Oxford University Press, 1992.

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

[Cha14] CHABERT G. : *IBEX 2.1.7 documentation*, 2014. `http://www.ibex-lib.org/doc/`.

[CJ11] CHABERT G., JAULIN L. : *QUIMPER, A language for Quick Interval Modeling and Programming in a Bounded-Error Context. User guide*, February 18, 2011. `http://www.emn.fr/z-info/ibex/site-archive/quimper_manual.pdf`.

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

[CM12] CIARLET P., MARDARE C. : On the Newton-Kantorovich theorem. *Analysis and Applications. Vol. 10*, Num. 3 (2012), 249–269.

[Coh93] COHEN H. : *A course in computational algebraic number theory*, vol. 138. Springer, 1993.

[Coh06] COHEN H. : Explicit methods for solving diophantine equations, 2006. Cours sur internet.

[COP12] COPRIN : ALIAS-C++, *A C++ Algorithms Library of Interval Analysis for equation Systems, version 2.7*, Septembre 2012. `http://www-sop.inria.fr/coprin/logiciels/ALIAS/ALIAS-C++/ALIAS-C++.html`.

[Fau99] FAUGÈRE J.-C. : A new efficient algorithm for computing Gröbner bases (F_4). *Journal of pure and applied algebra. Vol. 139*, Num. 1 (1999), 61–88.

[FM12a] FOUFOU S., MICHELUCCI D. : Bernstein basis and its application in solving geometric constraint systems. *Reliable Computing. Vol. 17* (2012).

[FM12b] FOUFOU S., MICHELUCCI D. : Interrogating witnesses for geometric constraint solving. *Inf. Comput.. Vol. 216* (2012), 24–38.

[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.

[Gal10] GALLIGO A. : Factorisation absolue de polynômes à plusieurs variables, chapitre du livre : Leçons de mathématiques d'aujourd'hui, volume 4, pp. 85–105, Cassini, Paris, présentées par F. Bayart et E. Charpentier, 2010.

[GB06] GRANVILLIERS L., BENHAMOU F. : Algorithm 852 : Realpaver : An interval solver using constraint satisfaction techniques. *ACM Trans. Math. Softw.. Vol. 32*, Num. 1 (mars 2006), 138–156.

[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.

[Imb13] IMBACH R. : *Résolution de contraintes géométriques en guidant une méthode homotopique par la géométrie*. Mémoire de thèse, Université de Strasbourg, 2013.

[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.

[JKDW01] JAULIN L., KIEFFER M., DIDRIT O., WALTER E. : *Applied interval analysis : with examples in parameter and state estimation, robust control and robotics*, vol. 1. Springer, 2001.

[JMS07] JERMANN C., MICHELUCCI D., SCHRECK P. : Modélisation géométrique par contraintes, chapitre du livre : Informatique graphique, modélisation géométrique et animation, Hermès science publications, éditeurs B. Péroche et D. Bechmann, 2007.

[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.

[KMF] KUBICKI A., MICHELUCCI D., FOUFOU S. : Witness Computation for Solving Geometric Constraint Systems. In *Science and Information (SAI) Conference 2014* (London Heathrow U.K), pp. 759–770.

[Mer00] MERLET J.-P. : ALIAS : an interval analysis based library for solving and analyzing system of equations. In *SEA* (2000), pp. 14–16.

[MF09] MICHELUCCI D., FOUFOU S. : Interrogating witnesses for geometric constraint solving. In *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling* (2009), ACM, pp. 343–348.

[MKC09] MOORE R. E., KEARFOTT R. B., CLOUD M. J. : *Introduction to Interval Analysis*. SIAM, 2009.

[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.

[MST*10] MICHELUCCI D., SCHRECK P., THIERRY S. E. B., FÜNFZIG C., GÉNEVAUX J. : Using the witness method to detect rigid subsystems of geometric constraints in CAD. In *Proc. of the 14th ACM Symposium on Solid and Physical Modeling, SPM 2010, Haifa, Israel, September 1-3, 2010* (2010), pp. 91–100.

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

[Sch03] SCHRIJVER A. : *Combinatorial optimization : polyhedra and efficiency*, vol. 24. Springer, 2003.

[SCL59] SKOLEM T., CHOWLA S., LEWIS D. : The diophantine equation $2^{n+2} - 7 = x^2$ and related problems. *Proceedings of the American Mathematical Society. Vol. 10*, Num. 5 (1959), 663–669.

[ST88] STROEKER R., TZANAKIS N. : On the application of Skolem's p-adic method to the solution of Thue equations. *Journal of Number Theory. Vol. 29*, Num. 2 (1988), 166 – 195.

[Thi10] THIERRY S. : *Décomposition et paramétrisation de systèmes de contraintes géométriques sous-contraints*. PhD thesis, Université de Strasbourg, Sep 2010.

[Tro09] TROMBETTONI G. : *Résolution de systèmes d'équations : l'essor de la programmation par contraintes sur intervalles*. Habilitation à diriger des recherches en informatique, Université de Nice-Sophia, Décembre 2009.