Re-parameterization reduces irreducible geometric constraint systems

Hichem Barki, Lincong Fang, Dominique Michelucci, Sebti Foufou

Re-parameterization reduces irreducible geometric constraint systems

Hichem Barki[a,*], Lincong Fang[b], Dominique Michelucci[c], Sebti Foufou[a]

[a] *CSE Department, College of Engineering, Qatar University, PO BOX 2713, Doha, Qatar*
[b] *School of Information Technology, Zhejiang University of Finance & Economics, 310018 Hangzhou, China*
[c] *LE2I UMR6306, CNRS, Arts et Métiers, Université Bourgogne Franche-Comté, F-21000 Dijon, France*

Abstract

You recklessly told your boss that solving a non-linear system of size n (n unknowns and n equations) requires a time proportional to n, as you were not very attentive during algorithmic complexity lectures. So now, you have only one night to solve a problem of big size (e.g., 1000 equations/unknowns), otherwise you will be fired in the next morning. The system is well-constrained and structurally irreducible: it doesn't contain any strictly smaller well-constrained subsystems. Its size is big, so the Newton-Raphson method is too slow and impractical. The most frustrating thing is that if you knew the values of a small number $k \ll n$ of key unknowns, then the system would be reducible to small square subsystems and easily solved. You wonder if it would be possible to exploit this reducibility, even without knowing the values of these few 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 methods relying on Hensel lifting) widely involved in geometric constraint solving and CAD applications can benefit from this decomposition with minor modifications. For instance, with $k \ll n$ key unknowns, the cost of a Newton iteration becomes $O(kn^2)$ instead of $O(n^3)$. Several experiments showing a significant performance gain of our re-parameterization technique are reported in this paper to consolidate our theoretical findings and to motivate its practical usage for bigger systems.

Keywords: Geometric Constraints Solving, Geometric Modeling with Constraints, Re-parameterization, Reduction, Decomposition

1. Introduction

Geometric modeling by constraints [1, 2, 3, 4, 5, 6, 7] leads to large systems of non-linear (algebraic most of the time) equations. In their seminal work, Gao et al. [8] automatically generated all the possible irreducible and structurally well-constrained 3D systems of geometric constraints (which they called basic configurations) that involve up to six geometric primitives (points, lines, and planes). These basic configurations correspond to 3D sub-problems that often occur in geometric constraint solving problems, in variational modelling, or in CAD/CAM applications. Most of the time, and contrarily to the 2D case, there is no closed-form solution for such 3D basic configurations. Gao et al. proposed the Locus Intersection Method (LIM) for solving these basic configurations and showed that among these possible 683 systems, 614 ones can be solved by using one key unknown (also called the driving parameter), while solving the remaining 69 ones requires two key unknowns. They referred to these two re-parameterization solving methods as LIM1 and LIM2.

Re-parameterization consists in identifying or introducing a small number of *key unknowns*, also called *parameters* in the literature (hence the re-parameterization term), which have the following property: *"if the values of these key unknowns were known, then the system would be reducible to much smaller structurally irreducible subsystems and thus it would be easily solved"*.

The work presented in [9] relied on properties of bipartite graphs underlying systems of equations, to polynomially decompose large systems into well-, over-, and under-constrained subsystems. The same paper [9] also proposed an efficient method to decompose or reduce well-constrained systems into irreducible and well-constrained (having as many equations as unknowns) subsystems. These decompositions have considerably speeded-up the solving process, and also allowed debugging systems of constraints in a constraint programming context. However, the reduction methods proposed in [9] have limitations. For instance, they do not apply to re-parameterized systems proposed in [8, 10, 11, 12]. This inability to reduce re-parameterized systems is due to the fact that the methods of [9] are unable to reduce irreducible systems, and that re-parameterized systems are irreducible. Later on, after the locus intersection method of Gao et al. [8] became popular, several techniques for the decomposition of geometric systems with re-parameterization have been proposed [12, 10, 11]. These methods decompose well-constrained 3D systems into re-parameterized subsystems with a small set of key unknowns per subsystem, perform in polynomial time, and provide sub-optimal but good results.

In spite of the breakthrough made by the re-parameterization

*Corresponding author
Email addresses: hbarki@qu.edu.qa (Hichem Barki), lincongfang@gmail.com (Lincong Fang), dmichel@u-bourgogne.fr (Dominique Michelucci), sfoufou@qu.edu.qa (Sebti Foufou)

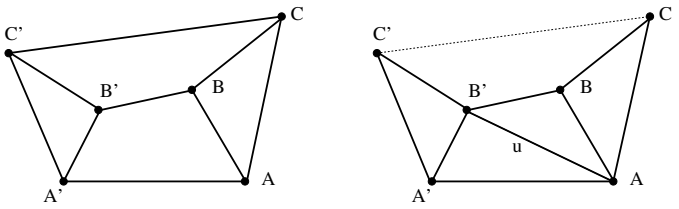

Figure 1: Left: a 2D system of geometric constraints, where each graph edge represents a distance constraint between two points corresponding to its incident nodes. For instance, edge AB implies that the distance AB is constant or fixed. Right: the re-parameterized system. If $u = AB'$ was known, then the system would be reducible and easy to solve, as the equation of the length of edge CC' would be redundant and could be ignored.

technique and the aforementioned methods seeking to find small sets of key unknowns, there are two major limitations. First of all, since re-parameterized systems are irreducible, the decomposition methods proposed in [9] cannot apply to them. Second, with basic configurations involving more than six geometric primitives or for systems of geometric constraints involving more complex geometric primitives (cylinders, spheres, cones, torii, etc.), using one or two key unknowns is not enough, even when employing the best re-parameterization techniques known so far [12]. In other words, although LIM1 (Locus Intersection method with one key unknown) is very fast and simple, its variants LIMk (Locus Intersection methods involving $k \geq 2$ key unknowns) become much less convenient.

Contribution. Our work addresses the aforementioned major limitations of re-parameterization/decomposition techniques. It proposes a technique for efficiently reducing or unlocking irreducible re-parameterized systems of equations like those proposed in [8, 10, 11, 12] and resulting from geometric constraint systems and geometric modeling applications, so that the decomposition methods proposed in [9], which were unable to reduce such re-parameterized irreducible systems, become applicable. Furthermore, this work shows that it is possible to benefit from these decomposition techniques even when the values of the key unknowns are not known and the number of these key unknowns k is greater than 2. For this purpose, we propose to exploit re-parameterization at the lowest level, which is the level of the underlying linear algebra routines: solving a linear system or inverting a matrix. The focus on the lowest level for exploiting re-parameterization is not hazardous. It is pertinent and highly motivated by the fact that most existing solvers, like Newton-Raphson or homotopy rely on the aforementioned low-level linear algebra routines. Consequently, this level seems to be the best place to exploit re-parameterization. Although, doing so doesn't prevent using re-parameterization at some higher level.

Our paper focuses on exploiting the re-parameterization technique for reducing and thus efficiently solving well-constrained irreducible re-parameterized systems which are determined in advance and for which the key unknowns or parameters (even if their values are unknown) are already identified. It doesn't seek to find the best decomposition or re-parameterization of a system, a problem that has already been investigated in the literature [12] (cf. section 10).

Although this paper focuses on solving well-constrained irreducible re-parameterized systems of equations and not on directly solving (under-constrained) geometric constraint systems involved in geometric modeling, the latter easily translate into well-constrained systems of equations which are perfectly handled by our technique. For instance, all the basic configurations enumerated in [8] can be solved by our technique which goes beyond the locus intersection method as the latter is limited to one or two key unknowns, while we do not have such limitation. Other examples of geometric constraint systems can be found in [12], while the particular case of the pentahedron problem and the way it is more efficienlty solved through re-parameterization are discussed in section 8.

The rest of this paper is organized as follows: we first introduce the re-parameterization technique in section 2 through examples, with a particular emphasis on the LIM involving one parameter. In section 3, we briefly present matching theory and show how combinatorial decomposition methods are applied in order to improve the performance of linear algebra routines. After that, we show in section 4 how decomposition speeds-up linear algebra routines. In section 5, we show how re-parameterization speeds-up linear algebra routines for re-parameterized systems, so that Newton and homotopy methods can straightforwardly benefit from re-parameterization. This section also draws a complexity study. Section 6 explains how Hensel lifting in p-adic methods can take advantage of re-parameterization as well. This is an important result as it shows that not only numerical analysis, but symbolic computations, like Gröbner bases, may also benefit from re-parameterization. Section 7 shows that interval solvers may also benefit from re-parameterization, however the wrapping effect requires further research. Section 8 presents an experimental study of the performance of our re-parameterization technique at the lowest level of linear algebra routines involved in numerous solvers (Newton-Raphson, homotopy, and also p-adic methods relying on Hensel lifting) and shows an important speed-up. This section also presents a CAD example showing the benefits of our re-parameterization technique when applied to geometric constraint systems. Section 9 examines the issues of using re-parameterization at a higher level. Finally, section 10 presents future works and open questions before section 11 concludes the paper.

2. Understanding re-parameterization

In this section, we first illustrate the re-parameterization technique by means of two examples in 2D and 3D.

2.1. A trivial 2D example

Fig. 1 depicts a system of geometric constraints in 2D. For this system, the lengths of all the edges are given. It is easy to see that this system is under-constrained because it involves twelve unknowns (2D coordinates of its six vertices) and nine (distance) constraints. To make this system well-constrained, we employ placement rules commonly used in the literature,

to constrain the placement of a particular subset of a geometric system [13], and transform it into a well-constrained system, without affecting the set of possible solutions. For our 2D system, we fix three coordinates in 2D, which is equivalent to fixing the positions of the three points of one triangle, say $A'B'C'$. Point $A'(x_{A'} = 0, y_{A'} = 0)$ is placed at the coordinates origin, point $B'(x_{B'} > 0, y_{B'} = 0)$ is placed on the positive x-axis, and point $C'(x_{C'}, y_{C'} > 0)$ is placed in the xy-plane with positive y coordinate. By doing so, the placement of triangle $A'B'C'$ has a unique solution which can be easily found, and this placement rule transforms the original under-constrained system into a well-constrained one of six equations (expressing the lengths of the six edges AB, BC, CA, AA', BB', and CC') in six unknowns (the coordinates of the vertices of triangle ABC, those of triangle $A'B'C'$ are known). This system has a finite number of solutions, at least for generic values of the lengths (e.g., non-null values). Unfortunately, this system is structurally irreducible, i.e., it is not decomposable into smaller well-constrained subsystems.

We observe that if the length u of edge AB' called a *key unknown* or *parameter* is known (cf. Fig. 1 right), then the whole system would be easy to solve. In fact, it would be easy to determine the coordinates of point A as a first step, by solving a system of two equations in two unknowns x_A and y_A. Geometrically speaking, A is one of two intersection points of two circles, the first circle has center A' and radius $l_{AA'}$ (length of edge AA'), while the second one has center B' and radius u. Then in a second step, the coordinates of point B can be computed in a similar way, as the intersection of two circles with known centers and radii. Third and finally, point C can be computed in three different ways because three equations are available (lengths of edges AC, BC, and CC'), but any of the three combinations of two of them is enough for determining point C. One of these equations can be "ignored", still assuming that u is known of course. We will assume that the ignored equation is the one corresponding to the length of CC'. So to summarize, if u is known, then the system becomes decomposable (or reducible) and thus very easy to solve.

The aforementioned reasoning holds when the value of u is known. However, u is in fact unknown. In order to solve the system for u, i.e., to compute the correct value of u, it is possible to plot the functions $A(u), B(u), C(u)$ (through sampling for example), and to detect when the ignored equation determining the length of edge CC' is satisfied. Note that by construction, all the other constraints are satisfied. Some iterations of the Newton-Raphson method or the dichotomy should be enough to tune the value of u. Since in some way, re-parameterization transforms an initial system of six equations in six unknowns into a system of one equation (the ignored equation) in one unknown (parameter u), it comes with no surprise that the re-parameterized system can be solved more efficiently than the initial one. However, many complications arise in the re-parameterized system due to the facts that: (1) $A(u)$, $B(u)$, and $C(u)$ are actually multi-functions (algebraically, square roots are involved; geometrically, two circles intersect in two points), and (2) the construction of $A(u)$, $B(u)$, and $C(u)$ may fail for some values of u (algebraically, for the square root of a negative number; geometrically, for the intersection of two disjoint circles). Finally, a third issue concerns the choice of the interval of the possible values of u. In this example, because of the triangle inequality of triangle $AA'B'$, u can be bounded as $u \leq l_{AA'} + l_{A'B'}$ and $u \geq |l_{AA'} - l_{A'B'}|$. Similarly, the triangle inequality of triangle ABB' implies that $u \leq l_{AB} + l_{BB'}$ and $u \geq |l_{AB} - l_{BB'}|$. When the computed intervals for the values of u are disjoint, there is no real solution for the system.

We define the *big re-parameterized system* as the concatenation of the initial system and another equation $u^2 - (A - B') \cdot (A - B') = 0$ that defines the key unknown u. The big system has one more equation and one unknown (u) than the initial system of six equations in six unknowns. It is also well-constrained as it has as many equations as unknowns. Its equations are independent (almost everywhere the Jacobian has a full rank), so the big re-parameterized system has a finite number of real solutions, at least for generic values of the given lengths.

Since u is an unknown of the big system, the latter is irreducible. Indeed, the equations of the big system are the following:

$$0 = \text{dist}^2(A, B') - u^2 \quad \text{equation defining } u \tag{1}$$
$$0 = \text{dist}^2(A', A) - l^2_{A'A} \tag{2}$$
$$0 = \text{dist}^2(B', B) - l^2_{B'B} \tag{3}$$
$$0 = \text{dist}^2(A, B) - l^2_{AB} \tag{4}$$
$$0 = \text{dist}^2(A, C) - l^2_{AC} \tag{5}$$
$$0 = \text{dist}^2(B, C) - l^2_{BC} \tag{6}$$
$$0 = \text{dist}^2(C', C) - l^2_{C'C} \quad \text{ignored equation,} \tag{7}$$

where $l_{A'A}, l_{B'B}, l_{AB}, l_{AC}, l_{BC}, l_{C'C}$ denote the lengths of the six edges and $\text{dist}^2(A, B) = (x_A - x_B)^2 + (y_A - y_B)^2$. The Jacobian of this system has the structure of the table depicted in Eq. 8, where the first column corresponds to the key unknown u, the last row corresponds to the ignored constraint, and an X is used in place of a non-zero entry.

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)

If the first column corresponding to u and the last row corresponding to the gradient of the ignored equation were removed, i.e., if the value of u was known so that the last equation becomes redundant and thus may be discarded, then the remaining system would be reducible. The order of unknowns and equations in the above system was deliberately chosen to emphasize and make visible this reduction in three blocks of two equations for each one, with a block lower triangular structure.

This development leads to the main question addressed in this work: *"is it possible to exploit re-parameterization, i.e., is it possible to reduce the big system in some way, even when the value of the key unknown* $\boldsymbol{u}$ *is indeed unknown?"* At first glance, it seems impossible to use decomposition because the big system is irreducible.

Remark 1. *Actually,* u *and its defining equation are useless.* x_A *can be used as the key unknown and the distance* $l_{C'C}$ *as the ignored constraint. Removing the first row (equation defining* u*) and the first column (derivatives w.r.t.* u*) results in the Jacobian with a visible block lower triangular structure in the upmost rightmost sub-square of the table depicted in Eq. 9, where the last row corresponds to the ignored constraint.*

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)

By symmetry, any of the unknowns x_A, y_A, x_B, y_B, x_C, and y_C can be used as the key unknown. However, rows and columns must be permuted for the block lower structure of the Jacobian to be visible.

2.2. A 3D example: the hexahedron

The main idea of re-parameterization is illustrated through the 3D example of the hexahedron problem (a generalization of the cube) in Fig. 2. The lengths of the twelve edges of the hexahedron are given as input. The system of constraints is completed with six coplanarity constraints (one for each face of the hexahedron), and a non-degeneracy constraint ensuring that the hexahedron is not flat. The last constraint is an inequality that eliminates a continuum of degenerate solutions of topological dimension 1, i.e., a curve of flat hexahedra [14]. To simplify, we will ignore this constraint in the sequel.

This system has a finite number of solutions, modulo rigid body motions, i.e., up to isometries. It has eighteen equations. The eight vertices are represented by 3×8 coordinates, six of which can be arbitrarily fixed as done for the 2D example in the previous section. For instance, vertex 0 is fixed at the origin, vertex 1 on the positive x axis, and vertex 2 on the Oxy plane with positive y coordinate. Thus $x_0 = y_0 = z_0 = y_1 = z_1 = z_2 = 0$. This placement rule yields a well-constrained system of eighteen (24-6) equations in eighteen unknowns. In addition, if we consider that $z_3 = 0$, which comes from the coplanarity of vertices 0, 1, 2, and 3, and constraint $x_1 = l_{01}$ (where l_{01} is the specified length for the edge 01 and x_1 denotes the abscissa of vertex 1), then we can satisfy and thus discard two additional constraints. At the end, we are left with a system of sixteen equations in sixteen unknowns, which is irreducible according to the methods proposed in [9].

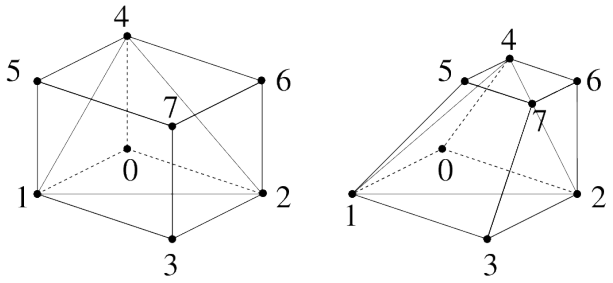

Figure 2: Two hexahedra. The values of the twelve hexahedron edge lengths are specified. The lengths of the three diagonal edges (triangle 124) are key unknowns. If these three values were known, then the problem would be reducible and easily solvable. In this case, the three length constraints of the edges incident to vertex 7 would be redundant and may be ignored.

The idea of re-parameterization when illustrated for the hexahedron problem is the following: if the lengths of the edges of the triangle numbered 124 are known, then it would be easy to compute the coordinates of triangle 012. Vertex 4 is the intersection of three spheres of known centers and radii. Then, it would be possible to compute the coordinates of vertex 3 as a function of the coordinates of vertices 0, 1, and 2. Similarly it would be possible to compute the coordinates of vertex 5 as a function of the coordinates of vertices 0, 1, and 4. It would also be possible to compute the coordinates of vertex 6 as a function of the coordinates of vertices 0, 2, and 4. Finally, the equations of the three planes 135, 236, and 456 can be computed and their intersection is vertex 7.

The distance constraints for edges 37, 57, and 67 have not been used as they are redundant. In this example, the key unknowns do not represent unknowns of the initial system, as they represent the lengths of the edges of triangle 124. The distance constraints for edges 37, 57, and 67 are called the *ignored* constraints. The Jacobian of the hexahedron has the structure depicted in table 1.

In some sense, re-parameterization transforms the initial system $S(X) = 0$ into a system $X = F(U), G(X) = G(F(U)) = 0$, where U are the key unknowns, $X = F(U)$ stands for the non-ignored constraints that are satisfied by construction, and $G(F(U)) = 0$ represents the ignored constraints. The unknowns of the *small re-parameterized system* are the parameters U and its equations are $G(F(U)) = 0$, unknowns X do not appear in this formulation. A difficulty arises unfortunately, F is a multi-function, e.g., it involves $\pm\sqrt{}$, and it may be undefined for some values of U. It may be difficult to find an explicit function F so that $X = F(U)$. In this case, one may prefer the implicit formulation $F(U, X) = 0, G(X) = 0$, or $F(U, X) = 0, G(U, X) = 0$, which is more general. It is the latter which is used in this paper, for the sake of generality.

When there is only one parameter U, computing the correct values for the parameter U is simple, because it reduces to following a parametric curve $X = F(U)$ or the implicit curve $F(U, X) = 0$, and to detect when the ignored constraint is satisfied (e.g., when the sign of $G(U)$ changes). Some iterations of

	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: The structure of the Jacobian of the big re-parameterized system for the hexahedron problem. Empty entries are zeros. The three first columns of the Jocobian matrix correspond to the key unknowns. The last three rows correspond to the ignored constraints. We set six unknowns to 0 as usual: $x_0 = y_0 = z_0 = y_1 = z_1 = z_2 = 0$. $D(A, B)$ is the distance constraint between vertices A and B. $C(A, B, C, D)$ is the coplanarity constraint of vertices (A, B, C, D). The lower triangular block structure of the upmost rightmost square is visible, thanks to the chosen permutations for rows (equations) and columns (unknowns).

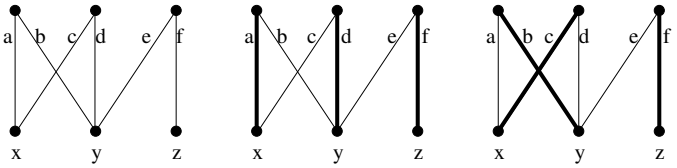

Figure 3: Left: the bipartite graph of a linear system $ax + by = r_1$, $cx + dy = r_2$, and $ey + fz = r_3$. Middle and right: the corresponding two perfect matchings adf and bcf. The determinant of the underlying matrix is $adf - bcf$.

the Newton method or the dichotomy allow to tune the solution values of U.

When U contains more than one parameter, the methods proposed in the literature to solve the small re-parameterized systems for U are much more difficult and more involved. In their pioneering work, Gao et al. [8] handle the case of one and two key unknowns and show that hundreds of irreducible 3D problems may be solved by using this re-parameterization. Our work presents a simple method, which applies for any number of key unknowns. The only restriction is that the number k of these key unknowns should remain small compared to the number n representing the size of the system to solve.

3. Matching theory and decomposition

The main motivation behind involving matching theory in our discussion comes from the fact that the decomposition of systems of equations mainly relies on this theory. In this paper, we briefly discuss this theory, while more details can be found in Lovasz and Plummer's book [15]. The reader is referred to this book for more information about matroids, König-Hall's marriage theorem, etc.

Each iteration of the Newton-Raphson method either performs a Jacobian matrix inversion, solves some linear systems, computes an LU decomposition, or performs similar linear algebra computations. Other solvers rely on similar linear algebra routines, like homotopy, also called continuation.

The methods proposed in [9] reduce well-constrained systems of (linear, or non-linear) equations to irreducible well-constrained subsystems. These methods are purely combinatorial, they study the bipartite graph of the equations and unknowns of the system. Each equation is represented by a graph vertex belonging to a first set of vertices, while each unknown is represented by a graph vertex belonging to a second set of vertices. An edge links an equation-vertex to an unknown-vertex if and only if the represented equation depends on that unknown.

The aforementioned decomposition methods apply to both linear and non-linear systems. They are based on maximum matchings. A *matching* is a subset of edges, so that each vertex in the graph is the vertex of at most one edge in the matching. A vertex which belongs to an edge in a matching is said to be *saturated*, or *covered*, by this matching. A matching is *maximum* if it is maximum in cardinality. A matching is *maximal* if it is maximum for inclusion, i.e., it is not included in a larger matching. Maximal matchings are not always maximum, but every maximum matching is of course maximal. A matching is *perfect* when all vertices are saturated. A perfect matching is both maximal and maximum.

The bipartite graph captures *structural properties* of the system. A system of non-linear or linear equations and its Jacobian matrix share the same bipartite graph. These properties depend only on that graph, regardless of the values of the coefficients of the system. For instance, the matrix has full rank (assuming generic values for the non-zero entries) if and only if the bipar-

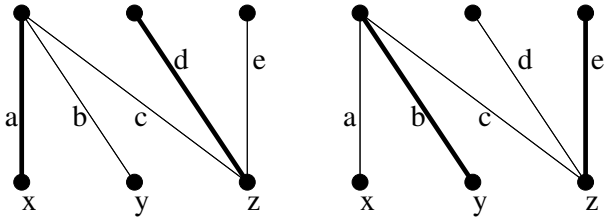

Figure 4: The depicted system $F_1(x, y, z) = F_2(z) = F_3(z) = 0$ is structurally not well-constrained. "Structurally" means that the system is not well-constrained regardless of the values of the coefficients a, b, c, d, and e. This is equivalent to say that its bipartite graph doesn't have any perfect matching. The maximum matchings which are ad, ae, bd, and be have cardinality 2, so the rank of the matrix is also 2 for generic coefficient values. represent the four possible maximum matchings of this graph.

tite graph has (at least) one perfect matching. More generally, the rank of the matrix, i.e., the number of independent equations of the system, cannot be greater than the cardinality of the maximum matching in the bipartite graph associated to this matrix. For some degenerate cases, the matrix rank is smaller than the cardinality of the maximum matching. This one-to-one correspondence between the terms of the determinant and the perfect matchings of the bipartite graph is illustrated in Fig. 3 for the matrix:

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

which has the determinant $adf - bcf$, while the perfect matchings are adf and bcf. The graph edges are labeled with the corresponding matrix coefficient. Note that in this example, the determinant is independent of the value of e and that the corresponding edge doesn't belong to any perfect matching.

This correspondence between a perfect matching and a determinant term holds for any square matrix. Indeed, let M be a square matrix and let G be the corresponding bipartite graph, i.e., G is the bipartite graph corresponding to the linear system $Mx = b$, where b is a given generic vector (non-null). An edge links the vertex representing the equation of the line l to a vertex representing an unknown c *if and only if* $M_{l,c}$ is non-null. Therefore, the determinant of M is $\sum_\sigma \text{parity}(\sigma) \prod_{l=1}^n M_{l,\sigma(l)}$, where σ runs through all permutations $1, \ldots, n$ and $\text{parity}(\sigma)$ is either $+1$ or -1 depending on whether the permutation σ is even or odd, respectively. We remind that a permutation σ is even (resp. odd) *if and only if* σ is the composition of an even (resp. odd) number of swaps of two distinct elements. Any non-null term $\prod_{l=1}^n M_{l,\sigma(l)}$ contains exclusively non-null $M_{l,\sigma(l)}$, so the permutation σ gives a perfect matching in G: this matching links the vertex of an equation l to a vertex of an unknown $\sigma(l)$.

This correspondence between a perfect matching and a determinant term explains why the determinant is null when there is no perfect matching as for the structurally not well-constrained system of Fig. 4. The qualifier *structurally* means that the system is not well-constrained regardless of the values of its coefficients a, b, c, d, and e.

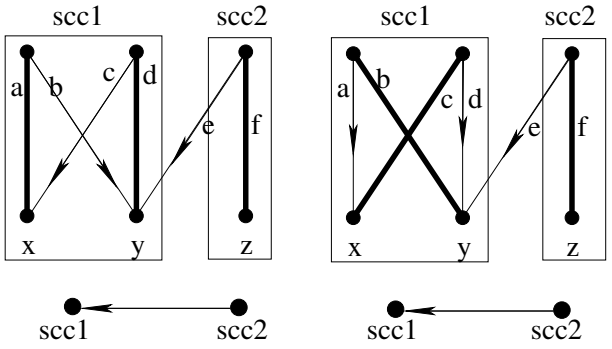

Figure 5: Upper row: the two perfect matchings of the system depicted in Fig. 3 and the corresponding strongly connected components induced in the oriented graph. The edges of the perfect matching (bold lines) are oriented in two directions, while the other edges are oriented downwards (from equations to unknowns). Lower row: in the reduced graph, each strongly connected component is reduced to a vertex. The edges of the reduced graph indicate the dependencies between the different subsystems. The reduced graph has no cycle.

The methods of [9] strongly polynomially (cf. [16] for the difference between strongly and weakly polynomial time) compute a maximum (perfect for well-constrained systems) matching of the bipartite graph corresponding to a system of equations. Then, they orient the edges of the bipartite graph according to whether they belong or not to the maximum matching. This process is illustrated in Fig. 5 for the example depicted in Fig. 3. For a well-constrained system, the only case discussed in this work, each strongly connected component of the oriented graph represents an irreducible well-constrained subsystem. The strongly connected components are independent of the used maximum matching. Moreover, the edges of the graph that link two distinct strongly connected components, i.e., the edges of the reduced acyclic graph, reflect the dependencies between the different subsystems, i.e., they provide the order for solving the different subsystems.

Some data structures are not only very convenient, but almost necessary in order to fully and easily benefit from the decomposition during the solving process, even for linear systems. These structures are the equation-unknown bipartite graphs and the Directed Acyclic Graphs (DAG for short) of subsystems, whose edges indicate the existing dependencies between the subsystems. These DAGs are the reduced graphs illustrated in Fig. 5. These data structures are even more indispensable in the case on non-linear systems, where we have to handle the multiplicity of the solutions of the non-linear subsystems, or on the other hand the absence of their solutions.

4. Reduction speeds-up linear algebra

The reduction or decomposition methods proposed in [9] reorder unknowns and equations in polynomial time, so that the Jacobian matrix becomes block lower triangular, and the decomposition into well-constrained subsystems becomes more visible. As a result, the processes of solving a linear system involving the Jacobian or inverting the Jacobian matrix become more efficient, thanks to the use of *forward block substitution*.

Remember that solving a linear system of n equations and n unknowns has a computational complexity of $O(n^3)$ in the general case by using Gauss pivoting, while solving a lower triangular system has a smaller complexity of $O(n^2)$ [17].

Let us assume that matrix M is block lower triangular as follows:

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

We will show how to exploit the decomposition of M in order to solve a system $MX = B$ or to invert matrix M. The proofs of what follows are straightforward and may be found in [17].

Remark that the matrices and the vectors mentioned in this section may either involve floating point arithmetic when manipulated by the Newton-Raphson or homotopy methods (section 5), interval arithmetic when manipulated by interval solvers (section 7), or rational/modular arithmetic when handled by some p-adic methods (section 6).

4.1. Inverting a block lower triangular matrix

The matrix K inverse of the square non-singular block lower triangular matrix M is also a block lower triangular matrix. It has the same block structure as matrix M

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

where its blocks $K_{l,c}$ are defined as:

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

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

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

In fact, it is always possible to avoid the inversion of matrix M as it suffices to just solve less than n linear systems (n is the number of rows or columns of M) as discussed in the following.

4.2. Solving a linear system

To solve a linear system $MX = B$, where $X = (X_1, X_2, \ldots)^t$ and $B = (B_1, B_2, \ldots)^t$ have a structure compatible with the block structure of matrix M, we have to successively solve the subsystems $X_1 := \text{solve}(M_{1,1}X_1 = B_1)$, then $X_2 := \text{solve}(M_{2,2}X_2 = B_2 - M_{2,1}X_1)$, then $X_3 := \text{solve}(M_{3,3}X_3 = B_3 - (M_{3,1}X_1 + M_{3,2}X_2))$, etc. In other words, we have to solve the series of the following l systems in ascending order of l:

$$X_l := \text{solve}(M_{l,l}X_l = B_l - \sum_{i=1}^{l-1} M_{l,i}X_i) \quad (16)$$

The smaller are the blocks $M_{l,l}$ and therefore numerous, the greater is the number of null blocks under the diagonal, and the more important is the speed-up in the solving process. In order to evaluate this speed-up, let us study the complexity for a simplified case, where M is of size n by n and all diagonal blocks are of equal size t by t. This implies that there exists $b = n/t$ diagonal blocks. If we denote by β the number of non-null matrix blocks below the diagonal, i.e., $0 \leq \beta \leq b(b-1)/2 \in O(b^2) = O(n^2/t^2)$, then solving $MX = b$ requires β products of a matrix of size t by t by a column vector of size t, which costs $O(\beta t^2)$, and the same number of additions of column vectors of size t, which costs $O(\beta t)$, and finally b times solving linear systems $M_{l,l}X_l = B_l - \sum_{i=1}^{l-1} M_{l,i}X_i$, $l = 1, \ldots b$, which costs $O(bt^3)$, at a cost of $O(t^3)$ per inversion (Strassen inversions are not pertinent for small matrices of size t by t when t is small). Therefore, the overall complexity is $O(\beta t^2 + bt^3) = O((\beta + bt)t^2)$. But since $bt = n$, the complexity becomes $O((\beta + n)t^2)$.

Now, let us assume that t is small as well, e.g., $t \leq 10$, so that it can be considered as constant $t \in O(1)$. The overall complexity of solving $MX = B$ is $O(\beta + n)$. We remind that $0 \leq \beta \leq b(b-1)/2 = O((n/t)^2) = O(n^2)$, so even if there is no null block below the diagonal, the complexity drops from n^3 to n^2. If β is of the order of n, then the complexity drops from n^3 to n. This may happen for geometric constraint systems because each constraint like the distance between two points depends on a constant number of variables. Therefore, the speed-up induced by the decomposition may be considerable.

In the following, we will put $\alpha = \beta + n$, where α is the number of non-null block matrices. We will say that the complexity of the solving process is $O(\alpha)$ and this holds because we assume that t is constant.

The optimal cost for solving a sparse linear system and the optional solving strategy (e.g., how to choose Gauss pivots that allow the most efficient system solving?) are discussed in more detail in Bomhoff's PhD thesis [18]. The exposed problems and the used combinatorial methods consider only the bipartite graph of the linear system.

5. Re-parameterization speeds-up linear algebra

This section shows that re-parameterization speeds-up linear algebra computations involved in the Newton-Raphson method or its variants, like for example the damped Newton method or the homotopy. In the sequel, we only consider the Newton-Raphson method.

We assume that the system of equations is well-constrained, the set of key unknowns is known, and the structure of the Jacobian matrix of the system has already been computed by the methods proposed in [9] and remains unchanged along the iterations of the Newton-Raphson method. Each Newton-Raphson iteration solves a linear system whose matrix has the structure (already seen in Eq. 8 or in the example of Fig. 1) depicted in Fig. 6.

Translating the Jacobian structure into formulas yields:

$$HU + AX = R, \quad CU + LX = S \quad (17)$$

where A is a square non-singular block lower triangular matrix, U is the vector of parameters or key unknowns, and X is the

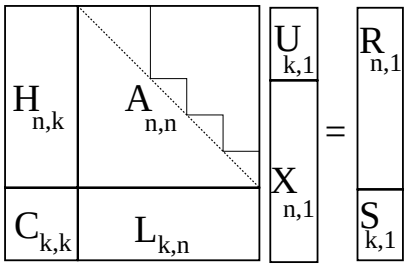

Figure 6: The structure of the Jacobian of a big re-parameterized system.

vector of the other unknowns. Eq. $HU + AX = R$ follows from the derivative of the non-ignored constraints, while Eq. $CU + LX = S$ follows from the derivative of the ignored constraints.

Let us now define the concept of an *R-structure*. Consider the following matrix extracted from the above equations:

$$\begin{pmatrix} H & A \\ C & L \end{pmatrix} \tag{18}$$

This matrix is said to have an *R-structure* (*R* for re-parameterization). Two matrices are said to have the same *R*-structure if and only if the sizes of their four blocks H, A, C, and L are equal, and the block structures of their A parts are equal.

From Eq.17, we deduce:

$$(C - LA^{-1}H)\,U \;=\; S - LA^{-1}R \tag{19}$$

which is proved as:

$$\begin{aligned} & (\text{Eq.17}) \Rightarrow CU + LX = S, \quad X = A^{-1}(R - HU) \\ \Rightarrow \quad & CU + LA^{-1}(R - HU) = S \\ \Rightarrow \quad & (C - LA^{-1}H)U = S - LA^{-1}R \end{aligned}$$

We will solve the last linear equation for U, then deduce X, avoiding at the same time the inversion of A. We assume that A is of size $n \times n$. The re-parameterization makes A highly reducible and this is the goal of re-parameterization. This decomposition of A allows to speed-up the solving process for linear systems $Ax = B$ as shown in the previous section. Solving $Ax = B$ (for a given B) costs $O(\alpha)$, i.e., the number of non-null blocks of matrix A. Equations 17 and 19 are solved for U and X as follows:

1 $Z_{n\times 1} := \text{solve}(A_{n\times n}Z_{n\times 1} = R_{n\times 1})$ costs $O(\alpha)$. This yields $Z = A^{-1}R$ which appears in Eq. 19, avoiding at the same time the inversion of matrix A.

2 $K_{n\times k} := \text{solve}(A_{n\times n}K_{n\times k} = H_{n\times k})$. This requires solving k linear systems, one for each column $c = 1, \ldots k$ of K, and the corresponding column of H. This costs $O(k\alpha)$. This yields $K = A^{-1}H$ which appears in Eq. 19, avoiding at the same time the inversion of matrix A.

3 Compute $C - LA^{-1}H = C_{k\times k} - L_{k\times n}K_{n\times k}$ costs $O(k^2 n)$. This term is involved in step 5.

4 Compute $S - LA^{-1}R = S_{k\times 1} - L_{k\times n}Z_{n\times 1}$ costs $O(kn)$. This term is involved in the right side of Eq. 19 and in step 5.

5 $U_{k\times 1} := \text{solve}((C - LA^{-1}H)_{k\times k}\,U_{k\times 1} = (S - LA^{-1}R)_{k\times 1})$, where only U is unknown, costs $O(k^3)$. This is Eq. 19. It is useless to optimize this step because k is small. This gives U.

6 Compute $R_{n\times 1} - H_{n\times k}U_{k\times 1}$ costs $O(nk)$. This term is involved in step 7.

7 $X_{n\times 1} := \text{solve}(A_{n\times n}X_{n\times 1} = (R - HU)_{n\times 1})$, where only X is unknown, costs $O(\alpha)$, avoiding at the same time the inversion of matrix A. This is the first equation of the system 17. This yields X.

Now, U et X are known. Consequently, the overall cost of solving a re-parameterized linear system is $O(k(\alpha + kn + k^2))$. As $k \ll n$ and because $n \leq \alpha \leq n(n-1)/2 \in O(n^2)$, this cost is therefore in-between $k^2(n + k)$ in the best case and $O(kn^2 + k^n + k^3) = O(kn^2)$ in the worst case. This cost is always less than n^3 which is the cost of solving a linear system by classical methods (Gauss pivoting or LU decomposition). It is even less than the cost of the inversion by the Strassen method which is $O(n^{\log_2 7}) \approx O(n^{2.8\ldots})$ and also less than the cost of the inversion by the Coppersmith-Winograd method which is $O(n^{2.375\ldots})$ (cf. [17] for more details, where it is shown that the inversion of a matrix of size $n \times n$ has asymptotically the same cost as the product of two matrices of size $n \times n$).

Another remark deserves to be mentioned. We have assumed for simplicity reasons that $k \ll n$ and we do not discuss the problem of determining the maximum value of k in order for the speed-up of the re-parameterization to remain interesting or significant. The aforementioned complexity study suggests a bound for k: if $k = O(n^{0.375\ldots})$, then our method has the same complexity as that of CoppersmithWinograd method which equals $O(n^{2.375\ldots})$. This bound for k is interesting because it is probably easier to find sets of key unknowns of size $O(n^{0.375\ldots})$ rather than sets of size $O(1)$ or $O(\log n)$.

6. Re-parameterization speeds up *p*-adic methods

p-adic methods are used in *p*-adic analysis, but also to solve diophantine problems [19, 20, 21, 22, 23] or in computer algebra computations, e.g., greatest common divisor of polynomials, polynomial factorization [24, 25], or Gröbner bases computation [26].

Re-parameterization can also benefit Hensel lifting used in *p*-adic methods. Let us assume that X_0 is a root of some re-parameterized algebraic system $F(X) = 0$ modulo p (a prime number or a power of a prime number). Our goal is to compute X_1 such that $X_0 + pX_1$ is a root of $F(X) = 0$ modulo p^2. In the Taylor series expansion $F(X_0 + pX_1) = F(X_0) + pF'(X_0)X_1 + \ldots$, we can clearly discard powers $p^k, k > 1$ (the dots at the end of the expansion) because they vanish modulo p^2. Then, $F(X_0) = 0 \mod p$ implies that $F(X_0)$ (taken modulo p^2) is a multiple of p. We compute the vector $\lambda = F(X_0)/p$. Then $F(X_0 + pX_1) = 0 \Rightarrow F(X_0) + pF'(X_0)X_1 = 0 \ [\mod p^2] \Rightarrow \lambda + F'(X_0)X_1 =$

$0 \pmod{p}$, which is a linear system, solvable modulo p. In fact, Hensel lifting is nothing else than the Newton-Raphson method but for the p-adics [26].

The proposed method can also be applied to compute the p-adic expansion of the root, i.e., the roots modulo $p, p^2, p^3, \ldots$ (or $p^2, p^4, p^8, \ldots$), starting from the root X_0 modulo p.

To summarize, if $F(x) = 0$ is a re-parameterized system, then each Hensel lifting may benefit from it. Note however that re-parameterization doesn't help to find the initial root X_0 modulo p.

7. Re-parameterization speeds up interval solvers

This section shows that interval solvers (*ALIAS-C++* [27], *RealPaver* [28], *IBEX* [29], and *QUIMPER* [30]) can benefit from re-parameterization. Interval solvers use methods of interval analysis [31, 32, 33, 34] that compute all the real roots of a system $f(x) = 0$ within a given initial box, where f denotes a large well-constrained re-parameterized system.

Typically, interval solvers handle a stack of boxes and work as follows: the initial box is pushed into this stack of boxes to be processed. While this stack is non empty, the top box B is popped and processed. This box processing tries to reduce the box without missing any root. One of the following cases may happen for a box B:

- Box B is proven to contain no roots (e.g., B is already an empty box).
- Box B is proven to contain only one regular root (the Jacobian is non-null), e.g., by some tests using Kantorovith theorem [35]. In this case, this unique root is tuned by some iterations of the standard (non-interval version) Newton-Raphson method and inserted into a list of roots.
- Box B is too small to be subdivided again and it is impossible to prove that it contains no roots. Box B may contain multiple roots, close roots, or no root at all, but the used arithmetic precision is insufficient. In this case, box B is inserted into a list of residual boxes.
- Box B has not been significantly reduced, so it may contain several real roots. In this case, B is bisected along its largest interval (other bisection choices have also been studied [27, 31, 32]) and the resulting boxes are pushed into the stack of non-processed boxes.

Several methods have been proposed in the interval analysis community for processing a box B, i.e., finding the roots of $f(x) = 0$ within B, or reducing B (maybe to the empty box) without missing any of its roots. One may look for the Krawczyck-Moore or the Sengupta-Hansen operators [32], just to cite few ones.

The simplest approach just solves with interval computations the linear system. If we denote by x_0 is the center of box B, then:

$$f(x \in B) = f(x_0 + (x - x_0)) \ \in \ f(x_0) + f'(B)(x - x_0) \quad (20)$$

We are looking for $x \in B$ solution of $f(x_0)+f'(B)(x-x_0) = 0$. Fig. 7 left provides a geometric interpretation of this equation for a 1D problem. Let us put $\Delta x = x - x_0$. Δx is the solution of the interval-wise linear system $f'(B)\Delta x = -f(x_0)$. $f'(B)$ is first computed by intervals, by exploiting the sparsity of the Jacobian f'. Then, we compute $\Delta x := \text{solve}(f'(B)\Delta x = -f(x_0))$ by exploiting the R-structure of the Jacobian, through the use of the of the methods described in section 5.

A main difficulty arises when any of the sub-matrices of the diagonal of the Jacobian $f'(B)$ is a singular matrix. This is the case of the 1D example of Fig. 7.

A solution to the aforementioned problem can be found by using the centered evaluation form. Let us consider $J = f'(B)$ and denote by J_0 the center of J, which is a non-singular matrix with probability one. It is also possible to use $J_0 = f'(x_0)$. Let us put $\Delta J = J - J_0$. It follows that:

$$\begin{aligned}
& f(x_0) + J\Delta x = 0 \\
\Rightarrow \quad & f(x_0) + (J_0 + \Delta J)\Delta x = 0 \\
\Rightarrow \quad & f(x_0) + J_0\Delta x + \Delta J\Delta x = 0 \\
\Rightarrow \quad & f(x_0) + J_0\Delta x + \Delta J(x - x_0) = 0, \ x \in B \\
\Rightarrow \quad & f(x_0) + J_0\Delta x + \Delta J(B - x_0) = 0 \\
\Rightarrow \quad & J_0\Delta x = -f(x_0) - \Delta J(B - x_0) \qquad (21)
\end{aligned}$$

This last equation represents a linear system having an unknown Δx. The elements of matrix J_0 are floating point numbers, not intervals. With probability one, J_0 is non-singular. In addition, J_0 has the R-structure of f, so solving the linear system can benefit from re-parameterization. Only the vector at the right side of the equation $-(f(x_0) + \Delta J(B - x_0)$ has interval elements. Geometrically, the hypersurface of each equation is bounded by a thick hyperplane of constant thickness, it is the convex hull of the cones of the previous linearization by intervals.

A simple 1D example is depicted in Fig. 7, where $B = [-2, 2]$, $x_0 = 0$, $f(x_0) = -1$, and $f'(B) = [-1/3, 2/3]$. The first linearization by intervals is $f(x_0+\Delta x) \in f(x_0)+f'(B)\Delta x$, which gives $-1 + [-1/3, 2/3]\Delta x = 0$. Geometrically, the arc of the curve is bounded by (or enclosed in) the two gray-shaded triangles, cf. left side of Fig. 7. $f'(B)$ is singular. The middle side of Fig. 7 shows the second linearization by intervals: the centered form. The arc of the curve is covered by a thick straight line, i.e., by the thick band of equation $J_0\Delta x = -f(x_0) - \Delta J(B - x_0)$. Numerically, $1/6x-[0, 2] = 0$. The right side of Fig. 7 superimposes the left and middle side coverings or bounds of the same figure. We observe that the intersection of the straight line Ox with the thick line is greater than its intersection with the two gray-shaded triangles. Fig. 8 illustrates a 2D example.

Once Δx is computed, box B is updated as $B := B \cap (x_0 + \Delta x)$. If this intersection is empty, then B doesn't contain any root. If the box is not reduced, then it is split into two boxes, e.g., using the largest side. Bisection is unavoidable because it is the only way to separate the roots.

The Gauss-Seidel idea can be used to optimize the computation of $B \cap (x_0+\Delta x)$: each i-th coordinate B_i of B may be reduced

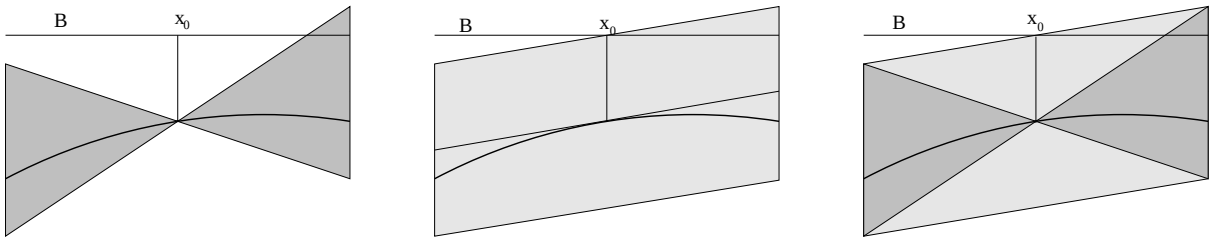

Figure 7: A function may be bounded in two ways by an interval-wise linear function. Left: $f(x) \in f(x_0) + f'(B)(x - x_0) = -1 + [-1/3, 2/3]x$ is the equation of the two gray-shaded triangles. Middle: $f(x) \in x/6 - [0, 2]$ is the equation of the gray-shaded band. Right: the left and middle figures superimposed, where the gray-shaded band is the convex hull of the two gray-shaded triangles.

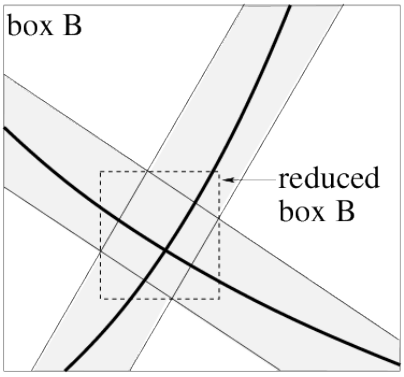

Figure 8: Linearization by intervals of a 2D system within a box B. The two involved curves are enclosed in two thick lines represented by the two gray-shaded bands. The solution of this 2D system is represented by the reduced box drawn in dashed lines.

with $B_i := B_i \cap (x_0 + \Delta x)_i$, as soon as it is available. The following computations use the latest and more precise value of B_i. If B_i is empty, then the processed box B contains no root. This idea is used in the Hansen-Sengupta operator.

Another classical optimization used in the Hansen-Sengupta operator is preconditioning, which limits the wrapping effect [36] of interval computations. The preconditioned system $g(x) = f'(x_0)^{-1} f(x) = 0$ is solved instead of the system $f(x) = 0$. By construction, $g'(x) = f'(x_0)^{-1} f'(x)$, so $g'(x_0)$ is the identity matrix. If f and g were linear, then the h-th hyperplane of equation $g'_h(x) = 0$ would be perpendicular to the axis x_h. The smallest is the box, the closer is the hypersurface $g'_h(x) = 0$ to a hyperplane. Is it possible to exploit both re-parameterization and preconditioning at the same time? The difficulty comes from the fact that g' doesn't have the same R-structure as f'. Indeed, the matrix product of two matrices having the same R-structure doesn't have the same R-structure in general.

To summarize, interval solvers may in principle benefit from re-parameterization, but further work has to be done to address the wrapping effect issue for re-parameterized systems.

8. Experimental results

We implemented our method for solving a linear re-parameterized system in Matlab and compared it with the naive Gaussian elimination method. We used the Matlab command $X = M \backslash B$ to solve the linear system $MX = B$ with the naive Gaussian elimination method. As motivated earlier in the paper, exploiting and thus benchmarking re-parameterization at the lowest level of the underlying linear algebra routines is highly motivated by the fact that most existing solvers (Newton-Raphson, homotopy, etc.) rely on these low-level linear algebra routines.

For the hexahedron problem, there is no significant difference between the two methods as the system is too small. Therefore, we have generated bigger random systems having the R-structure (section 5) and made a comparison.

For a system having the R-structure depicted in Figure 6, the performance comparison for the naive method and ours are provided in Table 2. In these experiments and for each combination of n (system size) and m (block size), we generated 100 random systems, solved them, and reported the average running time in milliseconds in the aforementioned table, where n is the size of the system, k is the number of key unknowns, and m is the size of each diagonal block. In other words, H is a $(n-k) \times k$ matrix, C is a $k \times k$ matrix, A is a $(n-k) \times (n-k)$ matrix, L is a $k \times (n-k)$ matrix, and each block of A, i.e., $A_{i,j}$ is a $m \times m$ matrix (cf. section 5 for the definition of matrices H, A, C, and L). The table column labeled "Naive" reports the solving timings for the naive Gaussian elimination method while the table column labeled "Re-param." reports the solving timings for our re-parameterization method. All non-zero entries of the involved matrices were generated randomly.

The empirical results clearly confirm that the re-parameterization method is significantly faster than the naive one and that it allows a more efficient solving for big systems of equations. For the reported experiments, our re-parameterization achieved an approximate $\times 26.6$ performance gain over the naive method. For $k = 5$ and $k = 10$, we obtained the same timings (and thus the same speed-ups) as those reported in table 2. These experiments emphasize another interesting fact about re-parameterization: the bigger are the systems of equations, the more important is the performance gain of the re-parameterization ($\times 3.7$ gain for systems with $n = 502$, $\times 8.1$ gain for systems with $n = 1002$, $\times 21.1$ gain for systems with $n = 2002$, and $\times 73.4$ gain for systems with $n = 4002$). This significant speed-up behavior shows the importance and the relevance of our re-parameterization method and highly motivates its adoption for bigger systems often encountered in practice.

No.	n	k	m	Naive (ms)	Re-param. (ms)
1	502	2	10	18.1	4.2
2	502	2	50	17.2	4.8
3	502	2	100	17.4	5.7
4	1002	2	10	71.5	7.6
5	1002	2	50	83.2	10.4
6	1002	2	100	83.5	12.0
7	2002	2	10	407.3	15.6
8	2002	2	50	426.8	20.1
9	2002	2	100	402.2	25.3
10	4002	2	10	2708.6	33.0
11	4002	2	50	2672.4	33.9
12	4002	2	100	2743.7	46.3

Table 2: Performance comparison of the naive method and our re-parameterization method for solving big linear systems. Significant speed-ups are achieved by our re-parameterization method over the naive one.

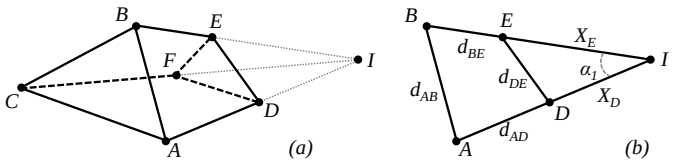

Figure 9: (a) The naive 3D pentahedron GCSP that yields an irreducible well-constrained system of 9 equations in 9 unknowns after using placement rules. (b) Re-parameterizing the pentahedron GCSP by introducing key unknowns representing the distances from 3 of its its vertices (composing one triangular face) to vertex I (intersection of its concurrent edges) results in a much smaller well-constrained system of 3 equations in 3 unknowns that is solved more efficiently.

Reducing the size of big irreducible subsystems indeed improves the performance of different solvers underlying CAD applications. So further to the aforementioned (linear algebra) low-level experiments, we demonstrate the benefits of re-parameterization technique by presenting the pentahedron problem [37, 38] that concretely shows how re-parameterization operates at the higher geometric level in order to reduce geometric constraint systems underlying CAD applications, and thus allows to solve them more efficiently.

A 3D pentahedron is a polyhedron composed of 6 vertices, 9 edges, and 5 faces, two of which are triangles and the remaining three are quadrilaterals, cf. Fig. 9a. The geometric constraint system traditionally associated to the 3D pentahedron is defined by the lengths of its edges and the planarity of its quadrilateral faces, translating into an under-constrained naive algebraic formulation of 12 equations in 18 unknowns, which can be reduced to a well-constrained irreducible system of 9 equations in 9 unknowns by using a placement rule that fixes one triangular face of the pentahedron in a plane, say $z = 0$, and thus fixes the 9 coordinates of its vertices, leaving the 9 coordinates of the other triangular face as system variables.

Even if the naive algebraic formulation of the pentahedron is simple, it is still difficult to solve by interval solvers, computer algebra (even when the distance parameters are instantiated with numeric values), and also Cayley-Menger determinants and their generalizations [39] that have been used to solve problems similar to the pentahedron (e.g., the octahedron GCSP). Most importantly, the pentahedron problem is unsolvable with only one key unknown by the re-parameterization technique proposed by Gao et al. [8] for solving 3D simple geometric problems similar to the 3D pentahedron.

Our re-parameterization of the pentahedron problem exploits a geometric property which is specific to non-degenerate pentahedra. This property states that the three supporting lines of the pentahedron edges AD, BE, and FG intersect at the same point I (Fig. 9a), which may be located at infinity if these lines are parallel (a case that is not considered in these experiments but which has been studied in [37]). Therefore, using this property, if we apply the "law of cosines" (cf. Fig. 9b) to obtain the equation of the angle α_1 of triangle ABI at vertex I, replace this equation into the equation of angle α_1 obtained similarly but by considering triangle DEI (both triangles lie on the supporting plane of the same quadrilateral face $ABED$ and share the same angle at vertex I), and do the same for the triangles and angles α_2 and α_3 defined by the pairs of triangles lying in the two supporting planes of the remaining two quadrilateral faces of the pentahedron, we obtain a new formulation/re-parameterization of the pentahedron GCSP described by Eq. 22 of Fig. 10, where variables X_D, X_E, and X_F denoting the distances from respective vertices D, E, and F to vertex I represent the key unknowns of our new well-constrained re-parameterized system of 3 equations in 3 unknowns, which is largely smaller than the naive one.

As a summary, by using re-parameterization, we reduced the pentahedron GCSP from an irreducible 9 equations/unknowns system into a simple 3 equations/unknowns system. Several experiments that we conducted in order to solve both systems (naive and re-parameterized) by the interval solver provided by *ALIAS-C++* interval analysis library [31] revealed a considerable performance enhancement (×42).

We shall note that the new formulation of the pentahedron GCSP is an *ad hoc* re-parameterization that doesn't generalize to other geometric problems. Nevertheless, this example gives an indication of the solving process performance improvements that the re-parameterization may bring when implementing the methods we propose in this work.

9. Re-parameterization at a higher level

Exploiting re-parameterization at the lowest level (of the widely used linear-algebra routines) has the advantages of simplicity and factorization, because it permits to reuse classical solving methods like Newton-Raphson, homotopy, and interval Newton methods at a small cost of minor implementation modifications or just by recompiling code, and makes these new re-parameterization-aware solver implementations a more efficient alternative to classical implementations underlying the solvers used in CAD applications, geometric modeling by constraints, or other fields. However, re-parameterization can also be used at a higher level. For instance, for an interval-based solver, it is possible to handle only boxes involving the parameters U, e.g., bisecting only boxes implying U. The boxes involving

$$\begin{cases} (X_D^2 + X_E^2 - d_{DE}^2)(X_D + d_{AD})(X_E + d_{BE}) - X_D X_E \left((X_D + d_{AD})^2 + (X_E + d_{BE})^2 - d_{AB}^2\right) = 0 \\ (X_E^2 + X_F^2 - d_{EF}^2)(X_E + d_{BE})(X_F + d_{CF}) - X_E X_F \left((X_E + d_{BE})^2 + (X_F + d_{CF})^2 - d_{BC}^2\right) = 0 \\ (X_F^2 + X_D^2 - d_{FD}^2)(X_F + d_{CF})(X_D + d_{AD}) - X_F X_D \left((X_F + d_{CF})^2 + (X_D + d_{AD})^2 - d_{CA}^2\right) = 0 \end{cases} \quad (22)$$

Figure 10: New re-parameterization of the 3D pentahedron GCSP that yields a well-constrained irreducible system of 3 equations in 3 key unknowns X_D, X_E, and X_F.

variables X are computed as a function of U through interval computations. The main difficulty at the higher level resides in the necessity for completely changing the implementation of the considered solver.

10. Preliminary results, and emerging questions

Many questions come into mind after analyzing the current work. The first one consists in whether interval Newton solvers can benefit from both re-parameterization and preconditioning, or from other ways of limiting the wrapping effect? For instance, among the many contraction operators available in interval analysis (Krawczyck-Moore operator, Hansen-Sengupta operator, etc.) or interval propagation operators (*Box consistency*: 2*B*, *Box*, *BC*4, *HC*4, *Mohc*, etc. [32]), is there any one that is suitable, or could be adapted to take advantage of re-parameterization? The same question arises for solvers based on tensorial Bernstein bases [40] or on Bernstein polytopes [41].

Another point concerns which redundant equations should be ignored? For example, one may think that it is better to ignore the highest degree equations.

For the sake of simplicity, we assumed that the number of key unknowns k is a constant or is smaller than n the number of unknowns and equations of the system. Presumably, a value of k of the order of $\log n$ or $\sqrt{n}$ remains interesting. In this sense, can we specify the maximum value of k that remains interesting?

Until now, we have not addressed the problem of computing a smaller set of key unknowns, whether for systems of equations or for geometric constraint systems. We remind that in [10, 12, 14], polynomial time algorithms have been proposed for geometric constraint systems. These algorithms compute small (may be not the smallest) sets of key unknowns. The generalization of these algorithms to more general (other than geometric constraint systems) sparse systems of equations is an open problem. In his PhD thesis [18], Bomhoff studied some similar problems related to systems of equations, for which the computation of the optimal solution is *NP*-hard, but the polynomial time computation of solutions close to the optimum is possible.

Similarly, one might think of the applicability of re-parameterization to other domains, like in linear algebra computations (SVD or QR decompositions, etc.), for solving linear programming problems, or for some symbolic computations.

Also for simplicity purposes, we have employed combinatorial methods for systems decomposition. Unfortunately, these methods detect only the structural dependencies and are unable to detect dependencies due to geometric theorems. Witness interrogation [42, 43, 44] allows to detect all the dependencies and also performs systems decomposition, assuming that a typical witness of the desired solution is known [45]. Hence the question: can we substitute the combinatorial methods by the witness method in order to take advantage from re-parameterization?

In this work, we focused only on well-constrained systems of equations, not on systems of geometric constraints. We have systematically transformed systems of geometric constraints into systems of equations, and then applied some placement rules (fixing three coordinates in 2D and six coordinates in 3D) in an ad hoc fashion in order to transform the original under-constrained geometric systems into well-constrained systems of equations. These placement rules depend on the coordinate system and do not constitute geometric constraints. This small difference between systems of equations and systems of geometric constraints may seem innocuous or insignificant at first glance. Unfortunately, it introduces many complications.

From a mathematical point of view, the combinatorial characterization, called the rigidity, of geometric constraint systems in 3D is still a research topic even for simple cases, where all constraints are generic distances between points. In 2D, the combinatorial characterization of the rigidity is given by Laman's theorem, but the latter is very restrictive, as for example, it assumes that all constraints are generic distances between points, excluding cocyclicity constraints, alignment constraints, etc.

In computer science, this difference complicates the decomposition of geometric constraint systems by combinatorial methods [5]. We may think that using the re-parameterization technique for geometric constraint systems is more difficult than using it for systems of equations, and that many works will be dedicated to these two questions.

11. Conclusion

In this work, we focused on reducing irreducible non-linear systems of equations, by exploiting the re-parameterization technique. The two main motivations of our work come from the inability of the decomposition methods proposed in [9] to reduce re-parameterized well-constrained systems of equations, and from the fact that for systems involving more than one parameter in their formulation, it is very difficult to exploit the re-parameterization technique [8, 10, 11, 12].

The current work generalizes the decomposition methods of [9] and makes them applicable to well-constrained

re-parameterized systems. We propose to exploit the re-parameterization technique at the lowest level of the omnipresent routines of linear algebra, so that many classical solvers, with a small cost of minor modifications, benefit from this technique and exhibit significant performance enhancements. The computational complexity gain may be very important. Therefore, our method opens the way to use the Newton-Raphson method and its variants (homotopy, interval solvers, etc.) for solving much larger systems with few key unknowns.

Acknowledgment

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.

References

[1] B. Brüderlin, D. Roller, Geometric constraint solving and applications, Springer, 1998.

[2] C. Hoffmann, R. Joan-Arinyo, A brief on constraint solving, Computer-Aided Design and Applications 2 (5) (2005) 655–663.

[3] C. Hoffmann, Summary of basic 2D constraint solving, International Journal of Product Lifecycle Management 1 (2) (2006) 143–149.

[4] B. Bettig, C. Hoffmann, Geometric constraint solving in parametric computer-aided design, Journal of Computing and Information Science in Engineering 11 (2) (2011) 021001.

[5] C. Jermann, G. Trombettoni, B. Neveu, P. Mathis, Decomposition of geometric constraint systems: a survey, International Journal of Computational Geometry & Applications 16 (05n06) (2006) 379–414.

[6] L. Garnier, H. Barki, S. Foufou, L. Puech, Computation of Yvon-Villarceau circles on Dupin cyclides and construction of circular edge right triangles on tori and Dupin cyclides, Computers & Mathematics with Applications 68 (12, Part A) (2014) 1689–1709.

[7] L. Garnier, H. Barki, S. Foufou, Dupin cyclide blends between non-natural quadrics of revolution and concrete shape modeling applications, Computers & Graphics 42 (0) (2014) 31–41.

[8] X.-S. Gao, C. Hoffmann, W.-Q. Yang, Solving spatial basic geometric constraint configurations with locus intersection, Computer-Aided Design 36 (2) (2004) 111–122.

[9] S. Ait-Aoudia, R. Jegou, D. Michelucci, Reduction of constraint systems, in: COMPUGRAPHICS, Alvor, Algarve, Portugal, 1993, pp. 331–340.

[10] A. Fabre, P. Schreck, Combining symbolic and numerical solvers to simplify indecomposable systems solving, in: Proceedings of the 2008 ACM Symposium on Applied Computing, 2008, pp. 1838–1842.

[11] R. Imbach, P. Mathis, P. Schreck, Tracking method for reparametrized geometrical constraint systems, in: 13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2011, pp. 31–38.

[12] P. Mathis, P. Schreck, R. Imbach, Decomposition of geometrical constraint systems with reparameterization, in: Proceedings of the 27th Annual ACM Symposium on Applied Computing, 2012, pp. 102–108.

[13] C. Durand, Symbolic and numerical techniques for constraint solving, Ph.D. thesis, West Lafayette, IN, USA (1998).

[14] R. Imbach, Résolution de contraintes géométriques en guidant une méthode homotopique par la géométrie, Ph.D. thesis, Université de Strasbourg (2013).

[15] L. Lovász, M. Plummer, Matching theory, North-Holland Mathematics Studies, Elsevier Science, 1986.

[16] A. Schrijver, Combinatorial optimization: Polyhedra and efficiency, Vol. 24, Springer, 2003.

[17] T. Cormen, C. Leiserson, R. Rivest, C. Stein, Introduction to algorithms, 3^{rd} edition, MIT Press, 2009.

[18] M. Bomhoff, Bipartite graphs and the decomposition of systems of equations, Ph.D. thesis, University of Twente, Enschede (2013).

[19] H. Cohen, Explicit methods for solving Diophantine equations, online lecture, http://swc.math.arizona.edu/aws/2006/06CohenLectures.pdf (2006).

[20] R. Stroeker, N. Tzanakis, On the application of Skolem's p-adic method to the solution of Thue equations, Journal of Number Theory 29 (2) (1988) 166–195.

[21] A. Baker, R. Plymen, p-adic methods and their applications, Oxford science publications, Clarendon Press, 1992.

[22] J. Box, An introduction to Skolem's p-adic method for solving Diophantine equations, Bachelor thesis, Korteweg-de Vries Instituut voor Wiskunde Faculteit der Natuurwetenschappen, Wiskunde en Informatica, Universiteit van Amsterdam (2014).

[23] T. Skolem, S. Chowla, D. Lewis, The Diophantine equation $2^{n+2} - 7 = x^2$ and related problems, Proceedings of the American Mathematical Society 10 (5) (1959) 663–669.

[24] A. Galligo, Factorisation absolue de polynômes à plusieurs variables, in: Lecons de mathématiques d'aujourd'hui, Vol. 4, 2010, pp. 85–105.

[25] H. Cohen, A course in computational algebraic number theory, Vol. 138 of Graduate Texts in Mathematics, Springer, 1993.

[26] J.-C. Faugère, A new efficient algorithm for computing Gröbner bases (F_4), Journal of pure and applied algebra 139 (1–3) (1999) 61–88.

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

[28] L. Granvilliers, F. Benhamou, Algorithm 852: Realpaver: An interval solver using constraint satisfaction techniques, ACM Transactions on Mathematical Software 32 (1) (2006) 138–156.

[29] G. Chabert, IBEX 2.1.7 documentation, `http://www.ibex-lib.org/doc/` (2014).

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

[31] J.-P. Merlet, ALIAS: an interval analysis based library for solving and analyzing system of equations, in: SEA, 2000, pp. 14–16.

[32] G. Trombettoni, Résolution de systèmes d'équations : l'essor de la programmation par contraintes sur intervalles, Ph.D. thesis, Université de Nice-Sophia (2009).

[33] R. Moore, R. Kearfott, M. Cloud, Introduction to Interval Analysis, Society for Industrial and Applied Mathematics (SIAM), 2009.

[34] L. Jaulin, M. Kieffer, O. Didrit, E. Walter, Applied interval analysis: with examples in parameter and state estimation, robust control and robotics, Vol. 1, Springer, 2001.

[35] P. Ciarlet, C. Mardare, On the Newton-Kantorovich theorem, Analysis and Applications 10 (3) (2012) 249–269.

[36] A. Neumaier, The wrapping effect, ellipsoid arithmetic, stability and confidence regions, in: Validation Numerics, Vol. 9, Springer Vienna, 1993, pp. 175–190.

[37] H. Barki, J.-M. Cane, L. Garnier, D. Michelucci, S. Foufou, Solving the pentahedron problem, Computer-Aided Design 58 (0) (2014) 200–209.

[38] H. Barki, J.-M. Cane, D. Michelucci, S. Foufou, 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, Vol. 8509 of Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 594–601.

[39] D. Michelucci, S. Foufou, Using Cayley-Menger determinants for geometric constraint solving, in: Proceedings of the ninth ACM symposium on Solid modeling and applications, 2004, pp. 285–290.

[40] S. Foufou, D. Michelucci, The Bernstein basis and its applications in solving geometric constraint systems, Reliable Computing 17 (2) (2012) 192–208.

[41] C. Fünfzig, D. Michelucci, S. Foufou, Nonlinear systems solver in floating-point arithmetic using LP reduction, in: 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling, 2009, pp. 123–134.

[42] D. Michelucci, S. Foufou, Interrogating witnesses for geometric constraint solving, in: 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling, ACM, 2009, pp. 343–348.

[43] S. Foufou, D. Michelucci, Interrogating witnesses for geometric constraint solving, Information and Computation 216 (0) (2012) 24–38.

[44] D. Michelucci, P. Schreck, S. Thierry, C. Fünfzig, J.-D. Génevaux, Using the witness method to detect rigid subsystems of geometric constraints in CAD, in: Proceedings of the 14th ACM Symposium on Solid and Physical Modeling, 2010, pp. 91–100.

[45] A. Kubicki, D. Michelucci, S. Foufou, Witness computation for solving geometric constraint systems, in: Science and Information (SAI) Conference 2014, London Heathrow U.K, 2014, pp. 759–770.