Over-constraints detection and resolution in geometric equation systems☆

Hao Hu, Mathias Kleiner, Jean-Philippe Pernot *

Arts et Métiers ParisTech, LSIS Laboratory, UMR CNRS 7296, France

ARTICLE INFO

Keywords:
Free-form surface deformation
Redundant and conflicting constraints
Structural decomposition
Numerical analysis
Decision support
Design intent

ABSTRACT

This paper proposes an original decision-support approach to address over-constrained geometric configurations in Computer-Aided Design. It focuses particularly on the detection and resolution of redundant and conflicting constraints when deforming free-form surfaces made of NURBS patches. Based on a series of structural decompositions coupled with numerical analyses, the proposed approach handles both linear and non-linear constraints. The structural decompositions are particularly efficient because of the local support property of NURBS. Since the result of this detection process is not unique, several criteria are introduced to drive the designer in identifying which constraints should be removed to minimize the impact on his/her original design intent. Thus, even if the kernel of the algorithm works on equations and variables, the decision is taken by considering the user-specified geometric constraints. The method is illustrated on academic and industrial examples realized with our prototype software.

1. Introduction

Nowadays, designers rely on 3D CAD software to model sophisticated shapes based on free-form curves and surfaces. In industrial design, this geometric modeling step is often encapsulated in a larger Product Development Process (PDP) which may incorporate preliminary design, reverse engineering, simulation as well as manufacturing steps wherein several actors interact [1]. Actually, the final shape of a product often results from a long and tedious optimization process which tries to satisfy the requirements associated to the different steps and actors of the PDP. Requirements can be seen as constraints. They are generally expressed either with equations, a function to be minimized, and/or using procedures [2]. The latter refers to the notion of black box constraints, not addressed in this paper, which focuses only on geometric constraints that can be expressed by linear or non-linear equations.

To satisfy the requirements, designers can act on variables associated to the different steps of the PDP. More specifically, in this paper, variables are supposed to be the parameters of the NURBS surfaces involved in the shape optimization process. To shape a free-form object defined by such surfaces, designers then have to specify the geometric constraints the object has to satisfy. For example, a patch has to go through a set of 3D points and satisfy to position constraints, the distance between two points located on a patch is fixed, two patches have to meet tangency constraints or higher-order continuity conditions, etc. Those geometric constraints give rise to a set of linear and non-linear equations linking the variables whose values have to be found. Due to the local support property of NURBS [3], the equations do not involve all the variables and some decompositions can be foreseen. Additionally, designers may express involuntarily several times the same requirements using different constraints thus leading to redundant equations. But, the designers may also involuntarily generate conflicting equations and may have to face over-constrained and unsatisfiable configurations.

Sometimes, over-constrained configurations can be solved by inserting extra degrees of freedom (DoFs) with the Boehm's knot insertion algorithm. As a consequence, many control points are added in areas where not so many DoFs are necessary [4]. This uncontrolled increase of the DoFs impacts the overall quality of the final surfaces which become more difficult to manipulate than the initial ones. Furthermore, some structural over-constraints cannot disappear following this strategy and dedicated decision-support approaches have to be developed to identify and manage over-constrained configurations.

Unlike advanced 2D sketchers available in most commercial CAD software, and which can interactively identify the over-constraints during the sketching process, it is not yet completely possible to pre-analyze the status of 3D NURBS-based equation systems before submitting them to a solver. Thus, there is a need for developing a new approach for the detection and

☆ This paper has been recommended for acceptance by Dr. Mario Botsch, Dr. Yongjie Jessica Zhang and Dr. Stefanie Hahmann.

* Corresponding author.
E-mail address: jean-philippe.pernot@ensam.eu (J.-P. Pernot).

resolution of redundant and conflicting constraints in NURBS-based equation systems. This corresponds to the identification and treatment of over-constrained, well-constrained and under-constrained parts. In this paper, the treatment corresponds to the removal of constraints before solving. Once the constraints removed, the equation system often becomes under-constrained and the designer also has to add a requirement by mean of a function to be minimized so as to solve and find the values of the unknowns. This aspect is not part of the proposed approach but it will be discussed when introducing the results in which a particular functional is minimized.

Removing user-specified constraints is a perious step as the result do not fully satisfies what the designers have specified. Thus, it is not only important to develop an approach that is able to remove over-constraints, but also desirable to develop decision-support mechanisms which can help the designers identifying and removing the right constraints, i.e. the ones which preserve as much as possible the initial design intent.

This work contribution is to address these two difficult issues by proposing an original decision-support approach to manage over-constrained geometric configurations when deforming free-form surfaces. The algorithm handles linear as well as non-linear equations and exploits the local support property of NURBS. Based on a series of structural decompositions coupled with numerical analyses, the method detects and treats redundant as well as conflicting constraints. Since the result of this detection process is not unique, several criteria are introduced to drive the designer in identifying which constraints should be removed to minimize the impact on his/her original design intent. Thus, even if the kernel of the algorithm works on equations and variables, the decision is taken by considering the geometric constraints specified by the user at a high level.

The paper is organized as follows. Section 2 introduces the background and reviews the related works. Section 3 introduces the framework of our algorithm, details the principles and characteristics of its different steps and proposes criteria for evaluating its results. The proposed approach is then validated on both academic and industrial examples which are described in Section 4. Finally, Section 5 concludes this paper by discussing the main contributions as well as future work.

2. Background and related work

This section introduces how designers can specify their requirements within an optimization problem. It also analyses the existing methods used to detect structural or numerical over-constraints.

2.1. Modeling multiple requirements in an optimization problem

During the last decades, many deformation techniques have been proposed and it is not the purpose of this paper to detail all of them. Most of the time, when speaking of deformation techniques working on NURBS curves and surfaces, the goal is to find the position X of some control points so as to satisfy user-specified constraints which can be translated in a set of linear and/or non-linear equations $F(X) = 0$. Since the problem is often globally under-constrained, i.e. there are less equations than unknown variables, an objective function $G(X)$ also has to be minimized. As a consequence, the deformation of free-form shapes often results from the resolution of an optimization problem:

$$\begin{cases} F(X) = 0 \\ \min G(X) \end{cases} \tag{1}$$

For some particular applications, the optimization problem can also consider that the degrees, the knot sequences or the weights of the NURBS are unknown. However, in this paper, only the position of the control points are considered unknown. Depending on the approach, different objective functions can be adopted but they often look like an energy function which may rely on mechanical or physical models. The constraints toolbox can also contain more or less sophisticated constraints with more or less intuitive mechanisms to specify them.

Thinking to the PDP as well as to the needs for generating shapes which satisfy multiple requirements, one can notice that designers have access to three main parameters to specify their requirements and associated design intent within an optimization problem. They can effectively act on the unknowns X to decide which control points are fixed and which ones can move. In this way, they specify the parts of the initial shape which should not be affected by the deformation. Of course, designers can make use of the constraints toolbox to specify the equations $F(X) = 0$ to be satisfied. Finally, designers can also specify some of their requirements through the function $G(X)$ to be minimized. For example, they can decide to preserve or not the original shape while minimizing an energy function characterizing the shape deformation.

However, most of the existing free-form shape deformation techniques do consider that the problem resulting from the set of equations $F(X) = 0$ is under-constrained [5,6] and few attention has been paid to the analysis and processing of possible over-constraints. This paper proposes an approach to detect conflicting and redundant equations, and to help the designer in solving those issues by simply removing some constraints. However, Sections 3.4 and 4 discuss the possibility to fix more or less control points and thus modify the unknown vector X, as well as the possibility to modify the overall deformation behavior through the customization of the objective function $G(X)$ to be minimized.

2.2. Geometric over-constraints

Geometric over-constraints are classified into structural and numerical over-constraints [7]. Structural over-constraints can be detected from an analysis of the DoFs, at the level of either the geometry or the equations. Numerical over-constraints are usually determined from an analysis of the solvability of the equations system. Since our approach is based on equations, both aspects are to be defined.

2.2.1. Structural over-constraints

Jermann et al. give a general definition of structurally over-constrained, well-constrained and under-constrained equation systems at a rather macro level and considering the dimension of the space [8]. This definition has been here adapted to system of equations where the system is expected to be fixed with respect to a global coordinate system.

Definition 1. The degree of freedom $DoF(v)$ of a geometric entity v is the number of independent parameters that must be set to determine its position and orientation. For example, in 2D space, it is equal to 2 for points and lines. For a geometric constraints system G with a set V of geometries, the degree of freedom of all the geometries is $DoFs = \sum_{v \in V} DoF(v)$.

Definition 2. The degree of freedom $DoC(e)$ of a geometric constraint e is the number of independent equations needed to represent it. For instance, distance constraints have one DoC in 2D and 3D. For a geometric constraints system G with a set E of constraints, the degree of freedom of all the constraints is $DoCs = \sum_{e \in E} DoC(e)$.

Definition 3. A geometric constraints system G is *structurally well-constrained* if G satisfies $DoCs = DoFs$ and if all the subsystems after decomposition satisfy $DoCs \leq DoFs$.

Definition 4. A geometric constraints system G is *structurally over-constrained* if there exists a subsystem satisfying *DoCs* > *DoFs*. *Structural over-constraints* are constraints that transform a *structurally over-constrained* system into a *structurally well-constrained* system when they are removed.

Finally, the above DoF-based counting definitions compare the number of equations to the number of variables of a system. However, they do not cover cases such as geometric redundancies induced by geometric theorems. In order to cover those situations, algebraic definitions are introduced.

2.2.2. Numerical over-constraints

The previous structural definitions cannot distinguish redundant and conflicting constraints. However, from the algebraic point of view, it is ascertain that this could be handled properly by Grobner basis or Wu-Ritt methods [9]. These methods are commonly used in abstract algebra and requires strong mathematical fundamentals to understand.

Definition 5. Let $G = (E, V, P)$ be a geometric constraints system, where E is a set of equations, V is a set of variables and P is a set of parameters. The set of solutions to G is denoted *Sol(G)*. A geometric constraints system is *inconsistent* iff $Sol(G) = \varnothing$ and is *consistent* iff $Sol(G) \neq \varnothing$.

Definition 6. Let $G = (E, V, P)$ be a *consistent* geometric constraints system. Let $G' = (E \cup E_c, V, P')$ be an *inconsistent* geometric constraints system, where E_c is a set of equations forming a constraint $C = \{E_c \mid E_c \cap E = \varnothing\}$ and $P \subset P'$. As a result, C is a *conflicting constraint* with respect to G.

Definition 7. Let $G = (E, V, P)$ be a *consistent* geometric constraints system. Let $G' = (E \cup E_r, V, P')$ be a *consistent* geometric constraints system, where E_r is a set of equations forming a constraint $R = \{E_r \mid E_r \cap E = \varnothing\}$ and $P \subset P'$, and *Sol(G)* is the same as *Sol(G')*. As a result, R is a *redundant constraint* with respect to G.

Definition 8. Let $G = (E, V, P)$ be a weakly connected geometric constraints system (its components are weakly connected [10]) which can be decomposed into the following two subsystems: $G_b = (E_b, V, P)$ and $G_o = (E_o, V, P)$ with $\{E = E_b \cup E_o, E_b \cap E_o = \varnothing\}$. If any constraint E_{oi} in E_o is either redundant or conflicting with respect to G_b, and if $\text{card}(E_b) \geq \text{card}(E_o)$, then E_b is a set of *basis constraints* and E_o is a set of *numerical over-constraints*.

It should be noted that the set of *basis constraints* and *numerical over-constraints* of a given system is not unique. Thus, decision-support mechanisms and criteria have to be defined to help designers identifying the right redundant and conflicting constraints to be removed.

2.3. Modeling geometric constraints system

As previously discussed, a geometric constraints system can be described either at the level of the equations or at the level of the geometry. On the one hand, for a system of equations, there exists algebraic methods able either to *directly* address consistency problems [11] or to analyze the structure *indirectly* based on a bipartite graph where two classes of nodes represent variables and equations independently [12]. On the other hand, for modeling at the level of the geometry, two types of graph are mostly used: either bipartite graphs with two classes of nodes representing geometric entities and constraints separately [13], or constraint graphs with nodes representing the geometric entities and edges representing constraints [14,15].

However, considering NURBS-based constraint systems is not straightforward since there exists several types of variables contributing to the shape deformation. On one hand, system of equations enable a sound way of modeling where parameters like knots and weights can be set as variables. On the other hand, graph modeling at the level of the geometry is currently limited to variables like the coordinates of the control points and their associated weights. Representing variables such as degrees, knots, values of u and v parameters using constraint graphs at the level of the geometry is not convincingly demonstrated in the literature. In the work of Lesage [16], vectors between control points are used to represent NURBS objects as well as the geometric constraints such as incidence or tangency. Nevertheless, his method is restricted to cases where control points are only unknowns and is not general enough comparing to equation-based modeling.

According to the previous definitions, detection methods can be classified in two categories [7]: structural over-constraints and numerical over-constraints detection.

2.4. Structural over-constraints detection

Many structural over-constraints can be identified while counting DoFs during the system decomposition process. Among the decomposition methods, those that are helpful for finding structurally over-constrained subparts have been identified and classified in three categories according to the type of subsystems.

2.4.1. Specific patterns

This category is based on the fact that, when considering their engineering drawings, most systems can be decomposed while recognizing specific patterns constructed by ruler and compass. Recursive division is first proposed by Owen to handle 2D constraint systems where only distance and angle constraints are involved [17]. He introduced several rules for redundancy detection, including rules to detect over-rigid subparts and rules to check angle redundancies [18]. Fudos and Hoffman adopted the bottom-up graph reduction method that works well with over- and well-constrained systems in two dimensions [19]. These methods are polynomial in time but not general enough due to the limited repertoire of patterns, which cannot cover all types of geometric configurations.

2.4.2. Structural rigidity

This class gathers together methods that decompose a system into structural rigid subsystems. Methods vary depending on the adopted structural-rigidity definition as well as on the corresponding search algorithms. By modifying incremental network maximum flow theory, Hoffmann et al. developed the *Dense* algorithm to identify 1-well-constrained subgraph [20]. Their definition of structural-rigid subsystem derives from Laman's theorem on the characterization of the rigidity of bar frameworks [21]. However, the definition does not handle properly constraints such as incidences and parallelisms, which are widely used in CAD systems. Jermann et al. modified their definition by introducing the notion of *degree of rigidity* (DoR) to replace the dimension D [22]. The difference between the two lies on the fact that the value of DoR varies with subsystems while D remains constant whatever the subsystems are (e.g. in 2D, $D = 3$ and in 3D, $D = 6$). Based on the same network flow theory, his algorithm *Over-rigid* deals properly with specified parallel/incidence constraints but is still restricted to generic conditions where incidence degeneracies due to geometric theorems like co-linearities, co-planarities are forbidden [23].

2.4.3. Maximum matching

Those methods recognize structural over-constrained patterns by directly comparing the DoFs and DoCs of a system (or subsystem) without considering the dimension-dependent constant D. The Dulmage–Mendelsohn (D–M) decomposition algorithm allows for decomposing a system of equations into over-constrained, well-constrained and under-constrained subsystems [24]. It has been used for debugging in equation-based modeling systems such as Modelica [12]. Additionally, it computes a directed acyclic graph (DAG) which provides a solving order among the strongly connected components (SCC) of the system. Serrano has been interested in using graph-theoretic algorithm to prevent over-constrained systems where all constraints and geometric entities are of DoF one [25]. Latham et al. extended the work of Serrano by proposing maximum b-weighted matching to identify over-constraints with arbitrary DoFs [26].

A structurally over-constrained part contains either redundant or conflicting constraints. But the remaining parts may also contain numerical over-constraints [27]. This is because structure-based analysis ignores numerical information of a system. Hence, to further identify subtle over-constraints like geometric redundancy, numerical methods have to be adopted.

2.5. Numerical over-constraints detection

Any geometric constraint can be transformed into a set of algebraic equations [13]. Therefore, geometric over-constraints are equivalent to a set of inconsistent (i.e. conflicting) or redundant equations. Here, numerical detection methods have been classified into two categories according to the type of constraints.

2.5.1. Linear over-constraints detection

Gauss elimination, LU factorization with partial pivoting and QR factorization with column pivoting have been successfully adopted to find redundant/conflicting equations as well as spanning group in systems of linear equations [28]. Light and Gossard applied Gauss elimination to compute the rank as well as to further identify invalid equations [29]. Serrano extended their work to check existence of over-constraints within strongly connected components of a system of equations [25]. These methods enable stable and fast detections but are limited to linear cases.

2.5.2. Non-linear over-constraints detection

Symbolic methods are theoretically reliable but the time complexity is exponential. Kondo used Grobner basis method to test dependency among 2D dimension constraints [30]. Gao and Chou introduced Wu-Ritt's decomposition algorithm to determine whether a system is over-constrained [14]. However, both methods do not directly find the spanning groups of over-constraints.

Optimization methods have been used to address constraints satisfaction problems, which works well for under-constrained systems [31].

Jacobian matrix analysis methods enable a faster detection by studying Jacobian structure of equations. However, they are not able to distinguish redundant and conflicting constraints. The main difference between these methods is the configuration where the Jacobian matrix should expand. If the system is solvable, Haug proposed to perturb the common root and recalculate the rank once the Jacobian matrix is rank deficient [32]. However, if the system is non-solvable, Foufou et al. suggest a Numerical Probabilistic Method (NPM), which analyzes the Jacobian matrix at random configurations [33]. However, there is a risk that the Jacobian matrix is rank deficient at the chosen points but is full rank everywhere else. Therefore, NPM is convenient in computation but may lead to incorrectly detected over-constraints. Instead of randomly selecting configurations, Michelucci et al. suggested to study the Jacobian structure at witness configurations where incidence constraints are satisfied [34]. The witness configuration and the target configuration share the same Jacobian structure. As a consequence, all the over-constraints are identified. More recently, Moinet et al. developed tools to identify conflicting constraints through analyzing the witness of a linearized system of equations [35]. Their approach has been applied to the well-known Double-Banana test case on which our approach will also be tested in Section 4.

From the above discussion, it is clear that different methods are capable of handling certain geometric constraints systems. However, no method is able to cover all cases perfectly within our criteria. Thus, in this paper, a new approach which couple structural as well as numerical methods is proposed.

3. A generic approach coupling structural decompositions and numerical analyses

This section describes our approach for detecting and treating redundant and conflicting geometric constraints. The main idea is to decompose the system of equations into smaller blocks that can be analyzed iteratively using dedicated numerical methods. The overall framework and algorithm are first introduced before detailing the different steps involved.

3.1. Overall detection framework

The overall framework has been modeled in Fig. 1. It is based on three nested loops: the structural decomposition into connected components (CC); the structural decomposition of a CC into its subparts ($G1, G2, G3$) and its corresponding DAG of strongly connected components (SCC); the iterative numerical analysis of these SCC. Pseudo-code for the main procedures is provided in Section 3.2.

3.1.1. Loop among connected components

The system of equations (SE) is initially represented by a graph structure G, where nodes correspond to variables and edges to equations. The structure is first decomposed into n connected components $\{CC_1, \ldots, CC_n\}$ using Breadth First Search (BFS) [36]. Such a decomposition is made possible thanks to the local support property of NURBS or simply when using constraints which decouple what happens along the x, y and z directions of the reference frame (e.g. position or coincidence constraints). As a result, geometric over-constraints can be detected separately for each CC_i.

3.1.2. Loop among subparts obtained by D–M decomposition

The D–M decomposition is used to structurally decompose CC_i into a maximum of three subparts: G_{i1} (over-constrained subpart), G_{i2} (well-constrained subpart) and G_{i3} (under-constrained subpart). Each subpart (if it exists) will be analyzed iteratively using the third nested loop explained below.

However, a single pass of the third loop on each G_{ij} is not sufficient. Indeed, any pass may lead to the removal of constraints, which modifies the structure of the CC_i and thus requires to apply D–M decomposition again after the pass to obtain updated subparts. The superscript d is used to note that CC_i^d (resp. G_{ij}^d) refers to CC_i (resp. G_{ij}) after its dth D–M decomposition. Although the number of passes required is unknown in advance, it is guaranteed that the process will converge to a state where only one subpart G_{i3} is left. In other words, constraints will be either removed or moved to the third subpart along the process.

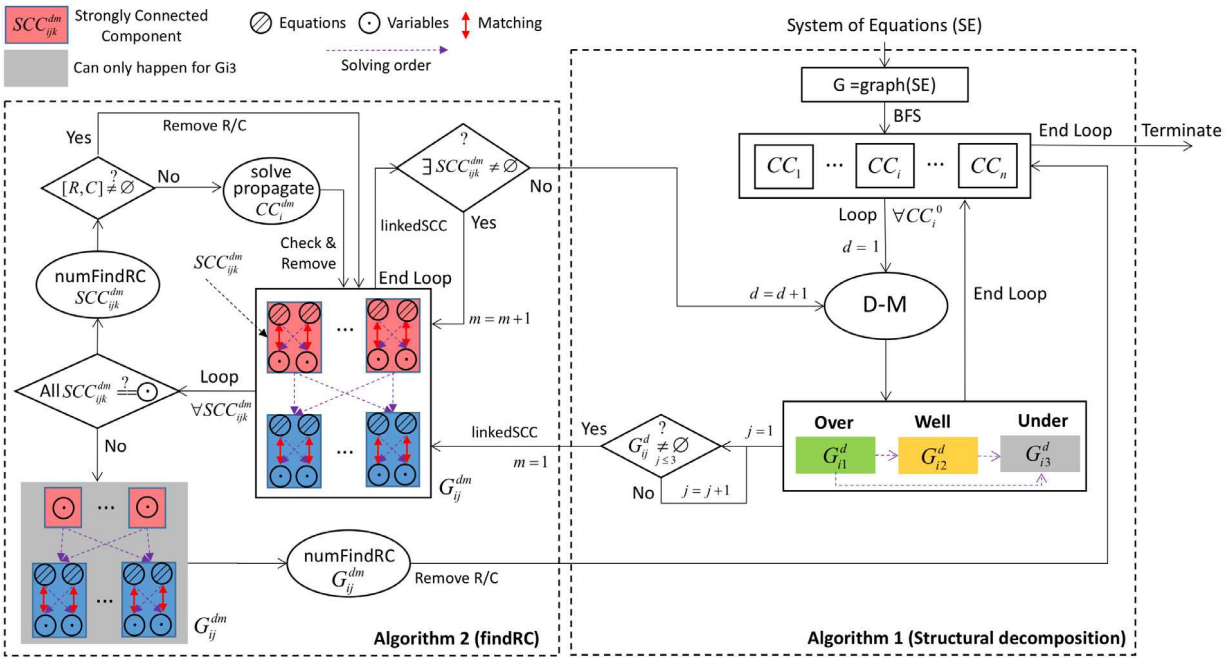

Fig. 1. Overall framework composed of three nested loops defining the main structure of the detection algorithm.

3.1.3. Loop among strongly connected components

In addition to the subparts, D–M decomposition also provides a DAG for each CC_i^d. Nodes of this DAG are strongly connected components SCC_{ijk}^d. Edges of this DAG (purple arrows in Fig. 1) denote solving dependencies between the SCC_{ijk}^d and may cross subparts G_{ij}^d boundaries. In the following, linkedSCC(G_{ij}^d) refers to the operation that obtains (the subpart of) this DAG from the dth D–M decomposition of CC_{ij}^d that corresponds to the given subpart.

The third loop consists in trying to iteratively (in the DAG-dependencies induced order) find numerical over-constraints in each SCC_{ijk}^d or, when it is solvable, propagate its solution to other blocks. Since blocks are strongly connected, there is only one potential solution to each block unless it contains only variables, and this latter case can only be encountered in a third subpart G_{i3}^d. The process works only the top-level of the DAG (red blocks in the figure) because these blocks equations do not use variables from other blocks.

For each red block, and as shown in the top-left part of the Fig. 1, an appropriate numerical method (numFindRC in the figure and the pseudo-code) tries to find redundant (R) or conflicting (C) constraints. These over-constraints are then removed from the currently analyzed connected component CC_{ij}^d. If the block is solvable, its (unique) solution is propagated to dependent blocks, which may lead to additional redundant or conflicting constraints being detected and removed from CC_{ij}^d. Once all red blocks have been analyzed, this part of the DAG (potentially turning blue blocks into red ones) is recomputed until all blocks are analyzed. However, it is not needed to recompute the D–M decomposition on the whole CC_{ij}^d, it is sufficient to recompute only the maximum matching for the current subpart by calling linkedSCC again. The superscript m is used to note that G_{ij}^{dm} (resp. SCC_{ijk}^{dm}) refers to G_{ij}^d (resp. SCC_{ijk}^d) after its mth matching. Although the number of passes required is unknown in advance, it is guaranteed that the process will converge to a state where there are either no blocks left, or these blocks only contain variables (and that is only possible for the third subpart G_{i3}^d). In other words, constraints and variables are removed until we obtain an under-constrained system with multiple solutions, meaning no more propagation is possible. In that last step, as shown in the bottom-left part of the figure, the remaining system is analyzed for numerical conflicts and proceeds with the next connected component.

3.2. Pseudo-code

This section provides the pseudo-code for the two main procedures of the approach, surrounded by dotted rectangles on Fig. 1.

Algorithm 1 Structural decomposition

```
SE ← System of Equations
G ← Graph(SE)
[CC_1, ···, CC_n] ←BFS(G)
for i = 1 to n do
    [G^1_i1, G^1_i2, G^1_i3] ←DM(CC_i)
    CC^1_i ← CC_i
    for j = 1 to 3 do
        d ← 1
        continue←True
        while continue and G^d_ij ≠ ∅ do
            continue,CC^{d+1}_i ← findRC(CC^d_i, G^d_ij)
            d ← d + 1
            [G^d_i1, G^d_i2, G^d_i3] ←DM(CC^d_i)
        end while
    end for
end for
return [CC^d_1, ···, CC^d_n]
```

3.3. Strongly connected components analysis

This section discusses the techniques used to analyze the strongly connected components SCC_{ijk}^{dm}. This corresponds to numFindRC function of Algorithm 2 (Section 3.2) used to find redundant (R) and conflicting (C) constraints of a component if they exist. Otherwise, the component is solved and the solutions are propagated to the whole system.

Depending on the type of constraints, i.e. either linear or non-linear ones, methods differ and are presented in the next subsections. The following notation $A[i : j, l : k]$ is used to define the matrix obtained by slicing the ith to jth rows, and the lth to kth columns of A.

Algorithm 2 findRC: Numerical analysis of G_{ij}^d subpart of CC_i^d

Require: CC_i^d and G_{ij}^d
Ensure: Boolean continue and updated CC_i^d
1: $[SCC_{ij1}^{d1}, \cdots, SCC_{ijN}^{d1}] \leftarrow$linkedSCC($G_{ij}^d$)
2: $m \leftarrow 1$
3: $G_{ij}^{d1} \leftarrow G_{ij}^d$
4: **while** $[SCC_{ij1}^{dm}, \cdots, SCC_{ijN}^{dm}] \neq \varnothing$ **do**
5: $l \leftarrow 0$
6: **for** $k = 1$ to N **do**
7: **if** onlyVariable(SCC_{ijk}^{dm}) **then**
8: $l \leftarrow l + 1$
9: **else**
10: $[R, C] \leftarrow$ numFindRC(SCC_{ijk}^{dm})
11: **if** $[R, C] == \varnothing$ **then**
12: solution $\leftarrow$ solve(SCC_{ijk}^{dm})
13: propagate(solution,CC_i^d)
14: $R \leftarrow$ checkRedundant(CC_i^d)
15: $C \leftarrow$ checkConflicting(CC_i^d)
16: **end if**
17: $CC_i^d \leftarrow$ removeRCfromCC(CC_i^d, $[R, C]$)
18: **end if**
19: **if** $l == N$ **then** ▷ all red blocks contain only variables
20: $[R,C] \leftarrow$ numFindRC(CC_i^d)
21: $CC_i^d \leftarrow$ removeRCfromCC(CC_i^d, $[R, C]$)
22: **return** False,CC_i^d
23: **end if**
24: **end for**
25: $G_{ij}^{d(m+1)} \leftarrow$ update(CC_i^d)
26: $m \leftarrow m + 1$
27: $[SCC_{ij1}^{dm}, \cdots, SCC_{ijN}^{dm}] \leftarrow$linkedSCC($G_{ij}^{dm}$)
28: **end while**
29: **return** True,CC_i^d

3.3.1. Linear system

In the proposed approach, the QR factorization with column pivoting is used to detect linear over-constraints. QR factorization with an optional column permutation P, triggered by the presence of a third output argument, is useful for detecting singularity or rank deficiency. Fig. 2 shows the overall detection process. The horizontal colored straight lines correspond to the linear equations of the system $A.x = b$ to be solved, where A has a dimension $m \times n$. Here, it is assumed that the rank of the system is r, which means that there are r independent equations with $m > r$. The rank is computed using SVD, which is relatively stable compared to other methods.

As for QR factorization, columns are exchanged at the start of the kth stage to ensure that

$$\left\| A_k^{(k)}(k:m) \right\|_2 = \max_{j \geq k} \left\| A_j^{(k)}(k:m) \right\|_2 \tag{2}$$

where $A_j^{(k)}(k:m) = A[k:m, j]$. At each step of the factorization, the column of the remaining un-factored matrix with largest norm is used as the basis for that step and is moved to the lead position [37]. This ensures that the diagonal elements of R occur in decreasing order and that any linear dependence among the columns is certainly revealed by examining these elements. Permutation matrix P rearranges the columns of A^t so that the columns appear in the decreasing order of their norm.

The first r columns of $A^t.P$ are the basis constraints of A^t and the first r columns of Q form an orthogonal basis (Fig. 2b). Since the remaining $m - r$ columns are linearly dependent on the first r columns [38], they are the over-constraints. The rank r also corresponds to the number of non-zero values of diagonal elements of R.

To find linear dependencies between the columns, the following deduction is needed. First, the matrix $Q(:, 1:r)$ is inverted using the following equation:

$$A^t(:, 1:r) = Q(:, 1:r).R(1:r, 1:r)$$

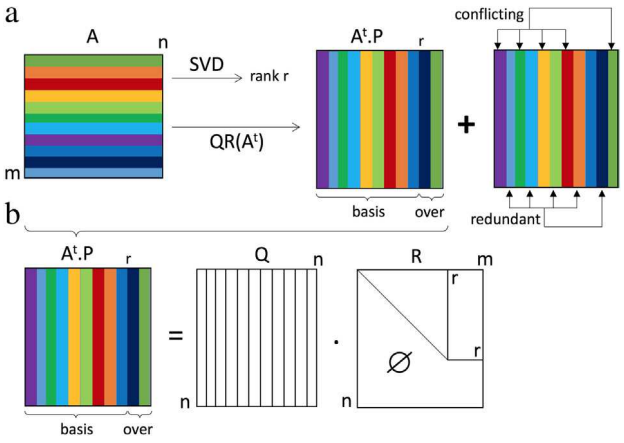

Fig. 2. Block analysis of linear systems: (a) overall detection process, (b) QR factorization with column pivoting.

and is then used in the following equation:

$$A^t(:, r+1:n) = Q(:, 1:r).R(1:r, r+1:n)$$

thus providing the following relationship between the two sliced matrices $A^t(:, r+1:n)$ and $A^t(:, 1:r)$:

$$A^t(:, r+1:n) = A^t(:, 1:r).R(1:r, 1:r)^{-1}R(1:r, r+1:n)$$

Finally, to identify the redundant and conflicting equations, the new b vector after factorization is redefined as follows:

$$b_{new} = b(r+1:n) - b(1:r).R(1:r, 1:r)^{-1}R(1:r, r+1:n)$$

Redundant and conflicting equations are further distinguished by comparing the value of the last $m - r$ elements of b_{new} with 0.

3.3.2. Non-linear system

When considering a system of non-linear equations, a two-phases identification process is used. First, the Witness Configuration Method [34] is used to find all the over-constraints (phase I), and Grobner Basis or Incremental Solving is then applied to further distinguish redundant and conflicting constraints (phase II).

Phase I. Taking advantage of the method proposed by Moinet et al. [35], a generic witness configuration is generated from the initial shape of the object to be deformed (step 1). Effectively, in our case, the variables x are the positions of the control points which have an initial location $x^{(0)}$ before deformation. Then, QR factorization with column pivoting is used to analyze this witness configuration (step 2). As a result, the sequence of equations is reordered. The first r (rank of the Jacobian matrix) equations are independent while the remaining ones are the over-constraints. In Fig. 3, the curved colored lines represent non-linear equations.

Phase II. To further distinguish redundant and conflicting constraints, either Grobner Basis or Incremental Solving are applied. In our algorithm, this choice relies on the number of equations. If the number of equations $m \leq 10$, Grobner Basis is preferred [39]. Otherwise, Incremental Solving is chosen. To explain the two methods, let us assume that the following constraints system is available after Phase I (Fig. 4):

$$\begin{cases} f_1(x_1, x_2, \ldots, x_n) = 0 \\ f_2(x_1, x_2, \ldots, x_n) = 0 \\ \vdots \\ f_r(x_1, x_2, \ldots, x_n) = 0 \\ f_{r+1}(x_1, x_2, \ldots, x_n) = 0 \\ \vdots \\ f_m(x_1, x_2, \ldots, x_n) = 0 \end{cases} \tag{3}$$

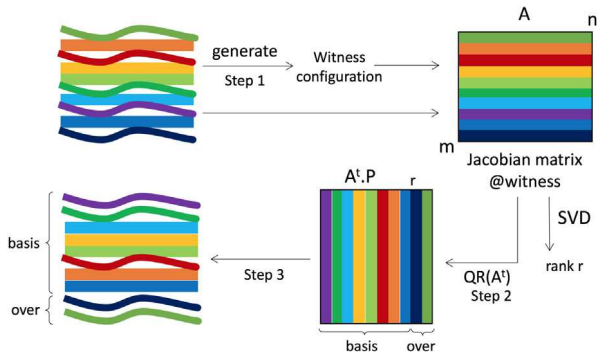

Fig. 3. Block analysis of non-linear systems. Phase I: Over-constraints detection.

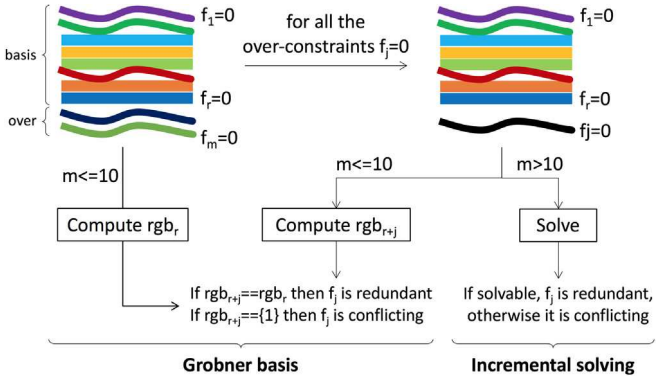

Fig. 4. Phase II: Distinguishing redundant and conflicting constraints.

where the Eq. (4) to r are the *basis constraints* and the equations $(r+1)$ to m are the *over-constraints*.

The Incremental Solving method incrementally insert the over-constraint $f_j = 0, j \in \{r+1, \ldots, m\}$, in the set of basis constraints thus forming a new group of equations $\{f_1 = 0, \ldots, f_r = 0, f_j = 0\}$. If the new group is solvable, then the equation $f_j = 0$ is redundant, otherwise it is conflicting. Of course, the basis constraints are always solvable.

When Grobner basis [11] are used, the method first computes the reduced Grobner basis rgb_r of the ideal $\langle f_1, \ldots, f_r \rangle$. Since the set of equations are solvable, $rgb_r \neq \{1\}$. Then, a loop on all the over-constraints $f_j = 0, j \in \{r+1, \ldots, m\}$, starts and for each over-constraint the reduced Grobner basis rgb_{r+j} of the ideal $\langle f_1, \ldots, f_r, f_j \rangle$ is computed. If $rgb_{r+j} \equiv rgb_r$, then $f_j = 0$ is a redundant equation. If $rgb_r \subset rgb_{r+j}$, then $f_j = 0$ is an independent equation. Finally, if $rgb_{r+j} = \{1\}$, then $f_j = 0$ is a conflicting equation.

3.4. Validation and evaluation of the solutions

Section 2.1 has introduced the multiple ways to model requirements within an optimization problem by specifying an unknown vector X, the constraints to be satisfied $F(X) = 0$ and the function $G(X)$ to minimize.

The approach described in this section allows for the identification of redundant and conflicting equations. Correctness is ensured since it consists of a fixed-point algorithm that only stops when the system is solvable. Additionally, any removed equation is guaranteed to be either conflicting or redundant with the remaining set. It has thus been shown that the set of equations $F(X) = 0$ can be decomposed in two subsets: $F_b(X) = 0$ containing the basis equations, and $F_o(X) = 0$ the over-constrained ones.

To stay close to the requirements the designer has in mind, the proposed approach then moves from the equations level to the constraints level. Thus, the geometric constraints associated to the equations $F_o(X) = 0$ are analyzed and all the equations related to those constraints are gathered together in a new set of equations $\widetilde{F}_o(X) = 0$. Of course, the equations $F_o(X) = 0$ are included in the set of equations $\widetilde{F}_o(X) = 0$. Finally, the equations related to constraints which are neither conflicting nor redundant form the other set $\widetilde{F}_b(X) = 0$. This transformation allows working at the level of the constraints and not at the level of the equations. This is much more convenient for the end-user interested in working at the level of geometric requirements.

Since this decomposition is not unique, it gives birth to various potential final solutions (interactive decomposition is out of scope of this paper). Therefore, several criteria are now introduced to evaluate these solutions according to the initial design intent. To be able to characterize the quality of the obtained solutions, the set of user-specified parameters P is introduced. This set gathers together all the parameters the designer can introduce to define the constraints his/her shape has to satisfy. For example, the distance d imposed between two points of a NURBS surface is a parameter characterizing a part of the design intent. Then, the idea is to evaluate how much the solutions deviate from the initial design intent and notably in terms of the parameters P.

To do so, the optimization problem containing the basis constraints is solved:

$$\begin{cases} \widetilde{F}_b(X) = 0 \\ \min G(X) \end{cases} \tag{4}$$

and the solution X' is then used to evaluate the unsatisfied over-constraints $\widetilde{F}_o(X')$ as well as the real values P' of the user-specified parameters P. For example, if the user-specified distance d between the two patches cannot be met, then the real distance d' will be measured on the obtained solution. From this solution, it is then possible to evaluate three quality criteria:

- *Deviation in terms of parameters/constraints*: this criterion aims at measuring how far/close the real values P' of the parameters are from the user-specified parameters P. This criterion helps understanding if the design intent is preserved in terms of parameters and consequently in terms of constraints.

$$df = \frac{\sum_i |P'_i - P_i|}{\sum_i |P_i|} \tag{5}$$

- *Deviation in terms of function to minimize*: this criterion directly evaluates how much the objective function G has been minimized. Here, the function is simply computed from the solution X' of the optimization problem. To preserve the design intent this value is to be minimized. Thus, it can be used to compare the solutions between them.

$$dg = G(X') \tag{6}$$

- *Degree of near-dependency*: rank deficiency of the Jacobian matrix at the witness clearly reveals the dependencies between constraints. However, for NURBS-based equation systems, the constraints can be independent but near to be dependent. In this case, the Jacobian matrix of $\widetilde{F}_b(X)$ at the solution point X' is ill-conditioned and the corresponding solution can be of bad quality. The third criterion thus evaluates the condition number (*cond*) of the Jacobian matrix as a measure of near-dependency [40]:

$$cond = cond(\mathbf{J}_{\widetilde{F}_b}(X')) \tag{7}$$

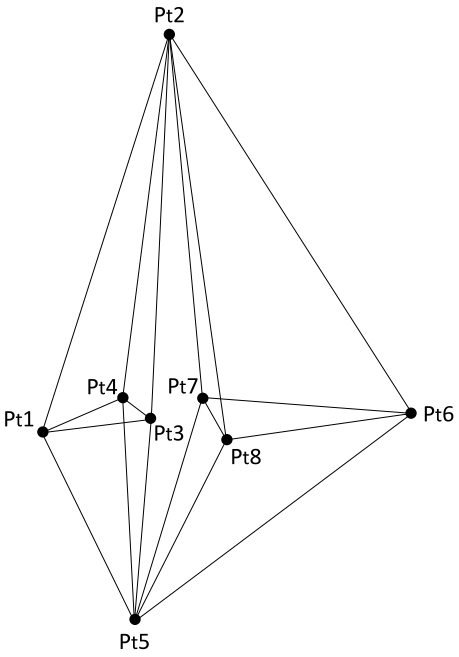

Fig. 5. Initial geometry of the Double-Banana as described in [35].

Finally, even if those criteria characterize the quality of the solution X' with respect to the design intent, they have not been combined in a unique indicator. Thus, the results of the next section will be evaluated by analyzing and comparing those three criteria for each solution.

4. Results and discussion

This section presents two configurations on which the proposed over-constraints detection and resolution technique has been tested. The first one concerns the academic Double-Banana testing case widely studied in the literature. It has been used to be able to compare our solution to the ones generated by others. The second example is more industrial and concerns the shaping of a glass composed of several NURBS patches.

4.1. Double-Banana testing case

The variables X, the constraints $F(X) = 0$, and the parameters P of the Double-Banana testing case are exactly the same as the ones tested by Moinet et al. [35]. The only difference is that they are using a coordinate-free formulation whereas ours is cartesian-based. Here, the objective is to find the position of the 8 nodes of a 3D structure so that the length of the 18 edges satisfy user-specified dimensions. Fig. 5 illustrates the Double-Banana in its initial configuration.

The Double-Banana configuration contains only one connected component as revealed by BFS. The structural analysis using D-M decomposition shows that it is under-constrained and our algorithm then follows the bottom part of the Fig. 1. WCM analysis is used within our numFindRC function and an over-constraint is detected. More specifically, the equation e9 is here detected. Using our Incremental Solving approach, the equation is further characterized as a conflicting one. The equation e9 is therefore removed and the system is solved using the initial position of the nodes as initial values of the variables. Using the results, the equation e9 is then reevaluated and the associated parameter is compared to the user-specified value. In the present case, e9 is not satisfied since it is equal to 44.47 compared to the initial user-specified requirement of 45. Thus, the deviation from the design intent is $df = 0.53/45$.

Table 1
Comparison between our algorithm and Moinet's approach on the Double-Banana testing case.

Method	Witness	Over-constraint	*df*	*cond*
Our	Initial sketch	e9	0.53/45	16.97
Moinet et al.	Initial sketch	e18	4.38/32	73.33

Our algorithm gives a solution that is much closer to the initial design intent than the algorithm of Moinet et al., and the remaining system is less ill-conditioned after removing the conflicting constraint (Table 1). Actually, the algorithm of Moinet et al. identifies e18 as a conflicting equation and its removal induces a deviation $df = 4.38/32$ from the initial design intent.

4.2. Sketching a 3D glass

In this example, the idea is to show how the proposed over-constraints detection and resolution approach can support the sketching of a 3D glass composed of 4 connected NURBS patches. The designer sketches his/her design intent and associated requirements. Here, the objective is to modify the upper part of the glass by specifying the following elements:

- *Variables*: Each patch has a degree 5×5 and has a control polygon made of 16×6 control points whose coordinates are the variables of our optimization process (Fig. 6a). Since the objective is to modify the upper part of the glass, the designer selects how many rows of control points are to be blocked and how many can move. For example, if the designer wishes to free the upper row of control points of the four patches, then there will be $6 \times 4 \times 3 = 72$ variables in the unknown vector X. The results will be illustrated with 4 and 5 rows free to move.
- *Constraints*: Three types of constraints are used to specify how the shape of the 3D glass has to evolve:
  - *Position*: 4 position constraints are added to the four end points of the patches along the upper boundary curves. As shown in Fig. 6c, the green points of the patches need to move to new positions in the 3D space. They are labeled from 1 to 4 and they generate $4 \times 3 = 12$ linear equations labeled from 1 to 12 (Table 2).
  - *Distance*: 2 distance constraints are defined between the opposite sides of the patches (Fig. 6d). They are labeled 5 and 6 and they generate 2×1 non-linear equations labeled 13 and 14 (Table 2).
  - *Coincidence and tangency*: 8 coincidence and 8 tangency constraints are specified to maintain the continuity between the upper parts of the patches during the deformation (Fig. 6b). They are labeled from 7 to 22 and they generate 8×3 linear and 8×3 non-linear equations labeled from 15 to 62 (Table 2). The non-linearity comes from the use of the vector product to express the collinearity of normals.

  Overall, there are 22 geometric constraints generating 62 equations in the set $F(X) = 0$. Some of those constraints are conflicting and it is the purpose of this section to try to see how our algorithm can detect and remove them without affecting too much the design intent.

Table 2
Typology of constraints and equations involved in the description of the 3D glass sketching example.

Constraint	Equations	Type	Component	Constraint	Equations	Type	Component
4	1–3	Linear	1	12	30-32	Non-linear	2
2	4–6	Linear	2	13	33-35	Linear	2
1	7–9	Linear	1	14	36-38	Non-linear	2
3	10–12	Linear	2	15	39-41	Linear	1
5	13	Non-linear	1	16	42-44	Non-linear	1
6	14	Non-linear	2	17	45-47	Linear	1
7	15–17	Linear	1	18	48-50	Non-linear	1
8	18–20	Non-linear	1	19	51-53	Linear	2
9	21–23	Linear	1	20	54-56	Non-linear	2
10	24–26	Non-linear	1	21	57-59	Linear	2
11	27–29	Linear	2	22	60-62	Non-linear	2

Table 3
Status of the distance and position constraints (0 to remove and 1 to keep) to solve the 9 over-constrained configurations.

Config.	DIS(P1,P4)	DIS(P2,P3)	POS(P1)	POS(P2)	POS(P3)	POS(P4)
1	0	0	1	1	1	1
2	1	0	0	1	1	1
3	1	0	1	1	1	0
4	0	1	1	0	1	1
5	0	1	1	1	0	1
6	1	1	0	0	1	1
7	1	1	1	1	0	0
8	1	1	1	0	1	0
9	1	1	0	1	0	1

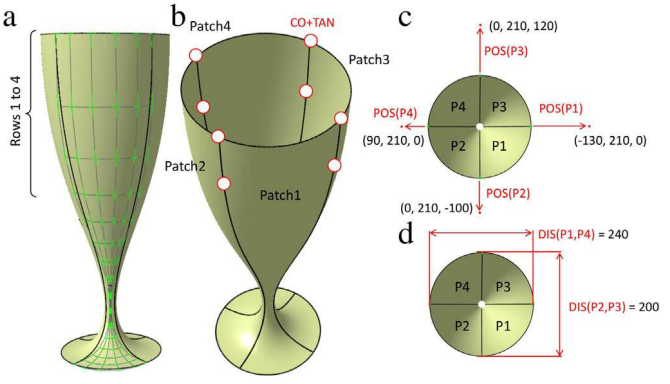

Fig. 6. Initial sketch of glass geometry.

- *Objective function*: Since the proposed approach removes the identified over-constraints, the resulting system of equations $\widetilde{F}_b(X) = 0$ (Section 3.4) may become under-constrained and a function $G(X)$ has to be minimized. Here, the idea is to make use of the approach of Pernot et al. to define two types of deformation behavior [4]: either a minimization of the variation of the shape ($G_1(X)$) between the initial and final configurations, or minimization of the area of the final shape ($G_2(X)$). In terms of design intent, the first one tends to preserve the initial shape of the glass, whereas the second forgets the initial shape and tends to generate surfaces similar to tensile structures.

As revealed by BFS, the system can be decomposed into $2 + 24 \times N_{rows}$ connected components CC_i where N_{rows} is the number of rows free to move. Among them, only two components CC_1 and CC_2 contain both variables and equations while the others contain only variables (Table 2). The analysis of those two components gives rise to the identification of 2 conflicting equations which correspond to either the position or distance constraints. Since the result of the detection process is not unique, 9 configurations are obtained and are gathered together in Table 3. Here, one has to remember that even if the detection process identifies conflicting equations, our algorithm removes the constraints associated to those equations. For example, configuration 1 considers that the two distance constraints (one between patches P1 and P4, and the other between P2 and P3) are to be removed (0 in the table) and the 4 position constraints are kept (1 in the table).

All configurations are then solved while acting on both the number of upper rows to be fixed ($N_{rows} = 4$ or 5), and the objective function to be minimized (either $G_1(X)$ or $G_2(X)$). The results are gathered together in Tables 4 and 5. Each configuration is evaluated through the three previously introduced criteria *dg*, *df* and *cond*. Some solutions are shown in Fig. 7.

One can first notice that depending on the configurations, the deviation *df* on the constraints varies. For example, with $N_{rows} = 4$ and while minimizing $G_1(X)$, the configuration 7 generates a solution that is closer to the design intent than configuration 6 ($0.10684 < 0.12607$ in Table 4). For configuration 3, it is clear that the deviation to the design intent in terms of constraints is more important when minimizing the area of the final surface than when minimizing the shape variation ($0.2288 > 0.10179$ in Table 4). This is clearly visible on Fig. 7 c_1 and c_2.

But the deviation dg_i on the objective function to be minimized also varies. While considering the minimization of the shape variation, one can see that configuration 3 is less interesting than configuration 1 in the sense that it minimizes less the shape variation ($15459.52 > 13801.04$ in Table 4).

Finally, for a given configuration, one can notice that when the number of free rows increases, i.e. when there is more freedom, the objective function decreases and the solution is therefore closer to the design intent. This is visible when comparing values from Tables 4 and 5. Thus, the selection of the variables X are also important when setting up the optimization problem.

5. Conclusion and future work

In this work, an approach for finding all over-constraints in free-form geometric configurations has been introduced. It relies on a coupling between structural decompositions and numerical analysis. The process and its algorithm have been described and analyzed with results on both academic and industrial examples.

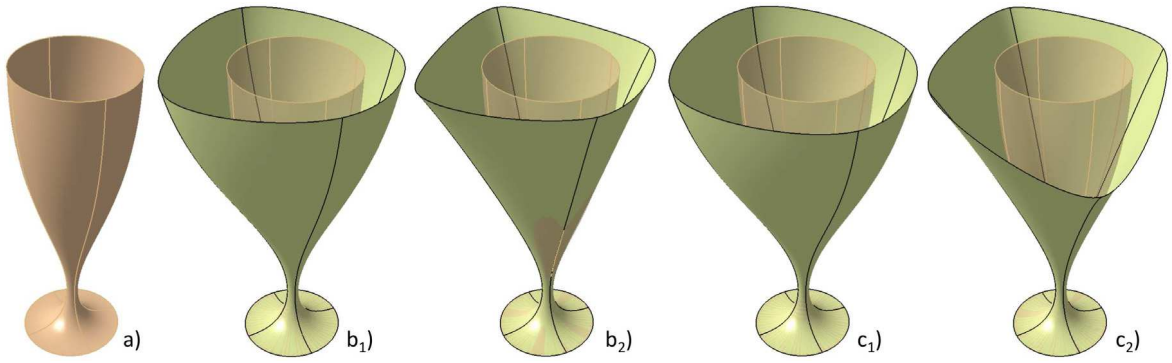

Fig. 7. Results of the sketching after removing conflicting constraints with $N_{rows} = 4$: (a) initial glass, (b_1) configuration 1 and minimization of the shape variation, (b_2) configuration 1 and minimization of the area of the final surface, (c_1) configuration 3 and minimization of the shape variation, (b_2) configuration 3 and minimization of the area of the final surface.

Table 4
Evaluation of the 9 configurations with $N_{rows} = 4$.

Config.	Minimization of $G_1(X)$			Minimization of $G_2(X)$		
	dg_1	df	$cond$	dg_2	df	$cond$
1	13801.04	0.10000	2.8654e19	96733.72	0.10000	1.4272e18
2	17990.88	0.10182	8.3172e18	95225.05	0.28157	4.3894e17
3	15459.52	0.10179	1.5071e19	94483.08	0.22880	4.9533e17
4	12265.51	0.10975	8.8857e18	89924.13	0.22806	3.9399e18
5	10970.98	0.10971	3.9852e19	86879.47	0.25225	8.2501e18
6	15826.68	0.12607	3.4260e18	76878.26	0.68278	1.6567e18
7	12936.45	0.10465	3.8205e18	76167.99	0.62820	3.9842e17
8	13889.18	0.11385	2.5681e18	78657.81	0.59160	4.1485e18
9	14883.21	0.11720	1.2523e18	74351.81	0.71765	6.7658e16

Table 5
Evaluation of the 9 configurations with $N_{rows} = 5$.

Config.	Minimization of $G_1(X)$			Minimization of $G_2(X)$		
	dg_1	df	$cond$	dg_2	df	$cond$
1	11266.93	0.10000	4.0149e17	85121.36	0.10000	3.3441e17
2	14719.05	0.10280	4.6031e17	86295.47	0.25034	6.5355e19
3	12506.55	0.10277	1.7748e19	85190.96	0.20076	1.0972e18
4	9944.87	0.11452	1.7903e18	79428.31	0.20592	1.7041e18
5	8799.29	0.11448	6.1454e17	77800.57	0.22919	1.0218e18
6	12561.66	0.13935	4.1681e18	69603.16	0.76646	8.5100e16
7	10441.11	0.10684	1.0862e18	69502.72	0.70009	2.3460e18
8	11134.09	0.12097	2.5394e18	71465.72	0.65901	1.5773e18
9	11877.59	0.12601	1.3790e19	67661.55	0.80372	8.4472e17

The approach has several benefits: it is able to distinguish between redundant and conflicting constraints; it is applicable on both linear and non-linear constraints; and it applies numerical methods on small sub-blocks of the original system, thus allowing to scale to some large configurations. Additionally, since the set of over-constraints of a system is not unique, it has been shown that our approach is able to provide different sets depending on the selected structural decomposition, and proposed criteria to compare them and assist the user in choosing the constraints he/she wants to remove. Even if the kernel of the algorithm works on equations and variables, the decision is taken by considering the geometric constraints specified by the designer at a high level.

A number of perspectives stem from this work. First, an automatization of the process should assist the designer in selecting the set of over-constraints that less deviate from his/her original design intent. As it is, the designer has access to three main criteria (dg, df, $cond$) which can be difficult to analyze for a non-expert. Thus, higher-level criteria should be imagined on top of those ones. Second, the approach can be made interactive, i.e. allowing the designer to select between the different conflicting sets along the process, or even modify the faulty constraints. Finally, it is planned to extend this work so that it can be used to detect and explain geometric configurations which, even when solvable, result in poor quality designs.

Acknowledgment

The authors are grateful to the China Scholarship Council (No. 201406090176) for supporting this research.

References

[1] Falcidieno B, Giannini F, Léon J-C, Pernot J-P. Processing free form objects within a product development process framework. In: Michopoulos JG, Paredis CJ, Rosen DW, Vance JM, editors. In: Advances in computers and information in engineering research, vol. 1., ASME-Press; 2014. p. 317–44.
[2] Gouaty G, Fang L, Michelucci D, Daniel M, Pernot J-P, Raffin R, et al Variational geometric modeling with black box constraints and DAGs. Comput Aided Des 2016;75:1–12.
[3] Piegl L, Tiller W. The NURBS book. In: Monographs in visual communication. Berlin, Heidelberg: Springer; 1996.
[4] Pernot J-p, Falcidieno B, Giannini F, Léon J-c. Fully free-form deformation features for aesthetic shape design. J Eng Des 2005;16(2):115–33.
[5] Elber G, Kim M-S. Geometric constraint solver using multivariate rational spline functions. In: Proceedings of the sixth ACM symposium on solid modeling and applications. ACM; 2001. p. 1–10.

[6] Bartoň M, Elber G, Hanniel I. Topologically guaranteed univariate solutions of underconstrained polynomial systems via no-loop and single-component tests. Comput Aided Des 2011;43(8):1035–44.
[7] Sridhar N, Agrawal R, Kinzel GL. Algorithms for the structural diagnosis and decomposition of sparse, underconstrained design systems. Comput Aided Des 1996;28(4):237–49.
[8] Jermann C, Trombettoni G, Neveu B, Mathis P. Decomposition of geometric constraint systems: a survey. Internat J Comput Geom Appl 2006;16(05n06): 379–414.
[9] Chou S-C, Gao X-S. Ritt-Wu's decomposition algorithm and geometry theorem proving. In: 10th international conference on automated deduction. Springer; 1990. p. 207–20.
[10] Diestel R. Graph theory. In: Electronic library of mathematics. Springer; 2006.
[11] Cox D, Little J, O'Shea D. Ideals, varieties, and algorithms: An introduction to computational algebraic geometry and commutative algebra. In: Undergraduate texts in mathematics. New York: Springer; 2008.
[12] Bunus P, Fritzson P. A debugging scheme for declarative equation based modeling languages. In: International symposium on practical aspects of declarative languages. Springer; 2002. p. 280–98.
[13] Hoffmann CM, Lomonosov A, Sitharam M. Geometric constraint decomposition. In: Geometric constraint solving and applications. Springer; 1998. p. 170–95.
[14] Gao X-S, Chou S-C. Solving geometric constraint systems. ii. a symbolic approach and decision of rc-constructibility. Comput Aided Des 1998;30(2): 115–22.
[15] Hoffman CM, Lomonosov A, Sitharam M. Decomposition plans for geometric constraint systems, part i: performance measures for cad. J Symbolic Comput 2001;31(4):367–408.
[16] Lesage D. Un modèle dynamique de spécifications d'ingénierie basé sur une approche de géométrie variationnelle. (Ph.D. thesis), Grenoble, INPG; 2002.
[17] Owen JC. Algebraic solution for geometry from dimensional constraints. In: Proceedings of the first ACM symposium on solid modeling foundations and CAD/CAM applications. ACM; 1991. p. 397–407.
[18] Owen JC. Constraints on simple geometry in two and three dimensions. Internat J Comput Geom Appl 1996;6(04):421–34.
[19] Fudos I, Hoffmann CM. A graph-constructive approach to solving systems of geometric constraints. ACM Trans Graph 1997;16(2):179–216.
[20] Hoffmann CM, Sitharam M, Yuan B. Making constraint solvers more usable: overconstraint problem. Comput Aided Des 2004;36(4):377–99.
[21] Graver J, Servatius B, Servatius H. Combinatorial rigidity. In: Graduate studies in mathematics. American Mathematical Soc.
[22] Jermann C, Neveu B, Trombettoni G. Algorithms for identifying rigid subsystems in geometric constraint systems. In: IJCAI, vol. 3. 2003. p. 233–8.
[23] Jermann C, Neveu B, Trombettoni G. Algorithms for identifying rigid subsystems in geometric constraint systems. In: IJCAI, vol. 3. 2003. p. 233–8.
[24] Dulmage AL, Mendelsohn NS. Coverings of bipartite graphs. Canad J Math 1958;10(4):516–34.
[25] Serrano D. Constraint management in conceptual design. (Ph.D. thesis), Massachusetts Institute of Technology; 1987.
[26] Latham RS, Middleditch AE. Connectivity analysis: a tool for processing geometric constraints. Comput Aided Des 1996;28(11):917–28.
[27] Podgorelec D, Žalik B, Domiter V. Dealing with redundancy and inconsistency in constructive geometric constraint solving. Adv Eng Softw 2008;39(9): 770–86.
[28] Strang G. Linear algebra and its applications. Thomson, Brooks/Cole; 2006.
[29] Light RA, Gossard DC. Variational geometry: a new method for modifying part geometry for finite element analysis. Comput Struct 1983;17(5-6):903–9.
[30] Kondo K. Algebraic method for manipulation of dimensional relationships in geometric models. Comput Aided Des 1992;24(3):141–7.
[31] Ge J-X, Chou S-C, Gao X-S. Geometric constraint satisfaction using optimization methods. Comput Aided Des 1999;31(14):867–79.
[32] Haug EJ. Computer aided kinematics and dynamics of mechanical systems. Allyn and Bacon Boston; 1989.
[33] Foufou S, Michelucci D. Interrogating witnesses for geometric constraint solving. Inform and Comput 2012;216:24–38.
[34] Michelucci D, Foufou S. Geometric constraint solving: the witness configuration method. Comput Aided Des 2006;38(4):284–99.
[35] Moinet M, Mandil G, Serre P. Defining tools to address over-constrained geometric problems in computer aided design. Comput Aided Des 2014;48:42–52.
[36] Leiserson CE, Schardl TB. A work-efficient parallel breadth-first search algorithm (or how to cope with the nondeterminism of reducers). In: Proceedings of the twenty-second annual ACM symposium on parallelism in algorithms and architectures. ACM; 2010. p. 303–14.
[37] Golub G, Van Loan C. Matrix computations. In: Johns Hopkins studies in the mathematical sciences. Johns Hopkins University Press; 2013.
[38] Dongarra J, Supercomputing S. Proceedings of the fourth SIAM conference on parallel processing for scientific computing. In: Proceedings in applied mathematics series. Society for Industrial and Applied Mathematics; 1990.
[39] Lamure H, Michelucci D. Qualitative study of geometric constraints. In: Geometric constraint solving and applications. Springer; 1998. p. 234–58.
[40] Kincaid DR, Cheney EW. Numerical analysis: mathematics of scientific computing. American Mathematical Soc.; 2002.