A Geometric Relaxation Solver for Parametric Constraint-Based Models

Lluis Solano Albajes
Pere Brunet Crosa
Universitat Politècnica de Catalunya
Departament de Llenguatges i Sistemes Informàtics
Secció d'Informàtica Gràfica
Av. Diagonal 647 planta 8 (ETSEIB)
08029 Barcelona. Spain

E-mail: solano@lsi.upc.es
E-mail: brunet@lsi.upc.es

Abstract

In this paper, a new relaxation algorithm for solving geometric constraint-based models is proposed. The algorithm starts from a constructive symbolic representation of objects (Constructive Parametric Solid Model, CPSM) and proceeds by iterative relaxation of the geometric constraints. Models that can be reduced to distance and angle constraints can be handled. A new algorithm based on an iterative global deformation of the system is presented and discussed, and its convergence is proved. The performance of hybrid algorithms involving global deformation and individual constraint relaxation is discussed on several practical cases.

1 Introduction

The term CAD is referred to the use of computers as an aid to the entire design process, including creation, modification, and visualization of designed parts. Traditional CAD-systems focus at the explicit object being designed. The construction process and the relationships between the involved elements are not reflected in the final design.

Constraint-based modeling allows the user to define families of object (generic objects) that can be subsequently converted into specific objects by giving the values of the generic object parameters and by solving the defined constraints.

The design process using CAD-systems is interactive: the user specifies step-by-step a sequence of operations that converges towards the final object. User interaction is performed through a Graphical User Interface. Modeling operations and geometric constraints can be chosen by the user in order to define the object. With the use of geometric constraints it is possible to specify the organization of a design. In this sense it is easy to generate design variations using constraints [Rol91]. Constraint-based design is aimed at representing and capturing the designer's intend. It allows to design generic objects more than explicit ones and even a family of designs instead of a single one.

The proposed constraint representation is based on the Constructive Parametric Solid Model (CPSM) wich is a procedural description of the modeling operations sequence and of the geometric constraints performed by the user during the interactive object design [SoB93].

The CPSM is the representation of a generic object of the whole family of objects, and it keeps the incremental design process. Previous defined models can be instanciated when a design is in progress; 2D, 3D and 2D_to_3D operations are supported. The CPSM is an Editable Representation (EREP) suitable for storage and transmission, it supports both generic and specific designs, and records the conceptual construction steps [RBN89] [HoJ93]. Each statement represents a modeling operation or a geometric constraint, expressed using a definition language [SoB94a]. In order to manage and keep the geometric constraints a specific structure called the Internal Model Representation is used.

In this paper, a new relaxation algorithm for solving geometric constraint-based models is proposed. The algorithm starts from a constructive symbolic representation of the objects (Constructive Parametric Solid Model, CPSM) and it proceeds by iterative relaxation of the geometric constraints. Models that can be reduced to distance and angle constraints can be handled. A new algorithm based on an iterative global deformation of the system is presented and discussed, and its convergence is proved. The performance of hybrid algorithms involving global deformation and individual constraint relaxation is discussed on several practical cases.

The paper is structured as follows: the next section presents the model that supports the constraints representation called the Internal Model Representation. After that, other geometric constraints approaches are reviewed with special emphasis on energy methods. Then the iterative global deformation is explained and proved. It is also discussed how to obtain models with distance constraints only from models with angle and distance

constraints. Finally, a constraint solver is presented based on an iterative relaxation method involving global and individual constraints.

2 The Internal Model Representation

The Internal Model Representation (IMR) is the structure used to manage and represent geometric constraints [SoB94b]. In the design process, the user can define dimensional constraints (distances and angles) between existing geometric elements. The IMR keeps explicit constraints, implicit constraints and default constraints that can be defined while the design is in progress. Constraint satisfaction uses the information stored in the IMR.

A set of basic geometric elements that can be used in modeling operations and constraints includes 0D elements (points), 1D elements (lines and edges), 2D elements (planes, polygons and circles) and 3D elements (polyhedra, etc). All of them must be instantiated and they are parametrically defined. They act as primitives within the final constructive parametric solid model. Modeling operations can either keep the dimension of the operands (for instance, in the case of boolean set operations between 3D elements) or increase it (like in sweep operations that transform a 2D element into a solid). Modeling operations can be parametric operations, that is, the result depends not only on the operands but also on the value of a number of formal parameters.

As a example, the CPSM description of the 2D L-shape polygon in figure 1-a would be,

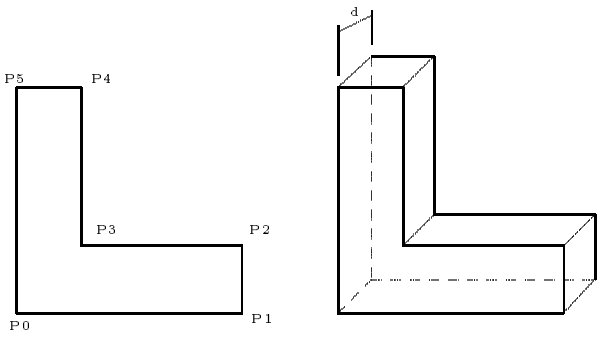

Figure 1: Example of 1D L-shape polygon

```
MODEL L_pol ( float d ) return model_id
{
   P1 := Point3D (0., 0., 0.)
   P2 := Point3D (0., 10., 0.)
   ...
   ...
```

```
   L_pol.OB1 := Closed_pol ( P1,P2,P3,P4,P5,P6, XY_REF())
   Dist_2P (P1,P2,d)
   Dist_2P (P2,P3,d/2)
   Dist_2P (P3,P4,d-d/2)
   Dist_2P (P4,P5,d-d/2)
   Dist_2P (P5,P6,d/2)
   Dist_2P (P6,P1,d)
   Dist_2P (P6,P3,d)
   Dist_2P (P5,P2,d)
   Dist_2P (P1,P4,sqrt(2)*d)
   RETURN L_pol.OB1
}
```

The polygon is modelled through a Closed_pol operation that constructs a new polygon on the first coordinate plane of the reference XY_REF(), from a given list of 3D points P1..P6, previously defined by the user through the user interaction. In the second part of the CPSM model description, a set of constraints that define the precise shape of the polygon are listed. As an example, Dist_2P(P6,P1,d) indicates that the distance between the bottom vertices of the polygon is d.

From a 2D polygon it is possible to generate a 3D object by sweep operation (figure 1-b). The next procedural description show the 3D extruded L-part:

```
MODEL L_part ( float d ) return model_id
{

   L_part.OB1 := Paral_sweep ( L_polygon (d), 0.5*d )
   return Mult_sweep (L_part.OB1)
}
```

A parallel sweep that instanciates the previous defined model of a 2D polygon is used in order to obtain the "L" shaped solid.

More precisely, the Internal Model Representation (IMR) can be defined as,

Definition 2.1. The Internal Model Representation is a graph $I = graph\langle N, C\rangle$ such that N is the set of nodes that represent points or vectors of the geometry and C is a set of edges that represent constraints between nodes(distances between points, angles between vectors). $C \subseteq N \times N$

Thus, the IMR keeps and manages constraints R between the existing geometric elements G.

If $\mathbf{P}$ is a point or vector of the model, then

$$\forall \mathbf{P} \in G \text{ if } \exists c \in R \mid c(\mathbf{P}) \Rightarrow \mathbf{P} \in N$$

A graph that represents the existing constraints between a set of elements is called *constraint graph*. It can be seen that all the modeling operations and constraints that are possible in the CPSM [SoB94a] can be represented as distances between two points and angles between two vectors. Geometric constraints in the CPSM can be therefore translated into distances and angles in order to be represented in the IMR graph.

Definition 2.2. A CPSM model with n points 2D is well-constrained iff there are $2n-3$ constraints and there doesn't exist any subgraph $G^{'}$ with $n^{'}$ points with more than $2n^{'}-3$ constraints [Lam70].

If a model is well-constrained with distance and angle constraints it is always possible to express the angle constraints as a function of a set of distance constraints (section 5). A model where the angle constraints have been converted to distance constraints is called a *d-reduced* model.

The IMR keeps the explicit constraints that have been defined by the user, the implicit constraints that reflect the relationships between geometric elements, such as a point that belongs to a polygon, and the default constraints which reflect the dimensions of the starting design of the product. The solver deals with this kind of constraints in a iterative way.

3 Geometric Constraint Solvers

In the literature there are different approaches to solve geometric constraints. There are two main strategies: the constructive one and the equational one. In the constructive approach, the constraints between geometric elements are solved in an order that reflects the construction process. Constructive methods are classified into:

- Procedural methods [Rol91a] [Rol91b] [RBN89] [CFV88] [Emm90] [SoB94b]
- Graph based methods [Owe91] [BFHCP93] [FuH93]
- Rule based methods [Ald88] [Sun88] [SoB91c] [VSR92] [JoS97]

In the equational methods, constraints are translated into a set of non-linear simultaneous equations. The system equations is solved by using different techniques such as:

- Newton-Raphson [LGL81] [Nel85] [SoB93]
- Geometric iteration [Bor81] [HsB97]
- Symbolic algebraic methods, as Gröbner basis [Buch88] or Wu-Ritt [Wen86] method
- Propagation methods [Sut63] [StS80]

In order to solve the distance constraints it is also possible to work in terms of energy minimization [WFB87] [KaB90]. In this case, constraints are relaxed in order to decrease the energy level using gradient techniques. The main problem is that the system can fall in local minima (see next section).

4 Energy Minimization Solvers

Energy models can be used in constraint satisfaction [WFB87] [KaB90]. In these approaches an energy continuous function is associated to each constraint. In this way the model has a certain energy level due to non solved constraints. The constraint satisfaction is performed by using an optimization which is usually based on the gradient method that calculates the new positions of the points in order to decrease the energy level of the model. A zero energy state means that the constraints have been satisfied. Thus, the energy approaches try to minimize an error function.

Given a 2D system with n points and m constraints $m = 2n - 3$, it is possible to state the following definitions:

State Vector $\vec{d}$: it contains the actual values of the parameters involved with the existing m constraints. $dim(\vec{d}) = m$

$$\vec{d} = \begin{bmatrix} d_1 \\ \vdots \\ d_m \end{bmatrix}$$

where d_k is a distance constraint between its endnodes $\mathbf{P}_{k_1}, \mathbf{P}_{k_2};\ k_1, k_2 \in K, \sharp K = n, m = 2n - 3$ in the plane.

Equilibrium Vector $\vec{d^c}$: it contains the target values of the parameters involved with the m constraints defined. $dim(\vec{d^c}) = m$

$$\vec{d^c} = \begin{bmatrix} d_1^c \\ \vdots \\ d_m^c \end{bmatrix}$$

Difference Vector $\vec{\delta}$: it is the difference between the state vector $\vec{d}$ and the equilibrium vector $\vec{d^c}$.

$$\vec{\delta} = \vec{d} - \vec{d^c}$$

Each constraint C_i between two points $\mathbf{P}_{i_1}$ and $\mathbf{P}_{i_2}$, has a constraint vector $\vec{F}_{jk}^{\,i}$ defined by,

$$\vec{F}_{12}^{\,i} = \delta_i \vec{u_i} = (d_i - d_i^c)\vec{u_i}$$

where $\vec{u_i} = \mathbf{P}_{i_1}\vec{\mathbf{P}}_{i_2} / ||\mathbf{P}_{i_1}\vec{\mathbf{P}}_{i_2}||$

Therefore, each constraint vector has associated an energy level that can be expressed as

$$E_i = \frac{1}{2}||\delta_i||^2 = \frac{1}{2}||\vec{F}_{12}^{\,i}||^2$$

The total energy level of the system is given by

$$E_T = \sum_{j=1}^{m} E_j$$

Every constraint defined in a model, has an actual value and a equilibrium value. The euclidean norm of the *difference vector* $\vec{\delta}$ is a measure of the energy level associated with the set of constraints.

Approaches based on energy models find a solution in under-constrained and well-constrained cases. In over-constrained cases these methods obtain a solution corresponding to a minimum energy level. The main problem arises when the system converges to a local minimum that is an unwanted solution. In these cases, [WFB87] proposes that the final solution is interactively decided by the user. If the system constraint values are compatible it can be guaranteed that the global minimum exists.

5 The Iterative Global Deformation Solver

This section is restricted to 2D systems involving only distance constraints. It is shown that a geometric relaxation algorithm can be derived such that the convergence to one of the solutions $\vec{d} = \vec{d}^{\,c}$ can be guaranteed. In other words, the relaxation solver converges to a global minimum $E_T = 0$ of the energy function. Convergence in the case of 2D systems involving distance and angle constraints will be discussed in the next section.

5.1 Relaxation of oriented constraints

The geometric relaxation algorithm is based on the following definition of oriented constraints:

Definition 5.1. For each distance constraint d_i^c defined between two points $\mathbf{P}_{i_1} = FirstPoint(d_i)$ and $\mathbf{P}_{i_2} = SecondPoint(d_i)$, the oriented constraint is defined as the 2D vector

$$\mathbf{R_i} = \frac{\mathbf{P}_{i_2} - \mathbf{P}_{i_1}}{d_i^c},\ i = 1 \ldots m$$

The set of all oriented constraints will be noted as $\{\mathbf{R}\}$. Therefore, $\{\mathbf{R}\} = \mathbf{R_1} \ldots \mathbf{R_m}$. Obviously, the set $\{\mathbf{R}\}$ has a total of $2m = 4n - 6$ degrees of freedom. The set of distances will be noted as $\{d\}$, $\{d\} = \{d_1 \ldots d_m\}$

Definition 5.2. The set $\{\mathbf{R_I}\}$ is the set of $n-1$ oriented constraints $\{\mathbf{R}\} = \mathbf{R_{k_1}} \ldots \mathbf{R_{k_{n-1}}}$ with indexes $k_1 \ldots k_{n-1}$ such that the set of model points $\{\mathbf{P}\} = \{\{\mathbf{P}_1 \ldots \mathbf{P}_n\}$ can be computed from $\{\mathbf{R_I}\}$.

(Without loss of generality, it is next assumed that $\mathbf{P}_1 = (0,0)$ and $(\mathbf{P}_2 - \mathbf{P}_1)^T \cdot (0,1) = 0$).

Lemma 5.1. *In a well constrained 2D model with distance constraints, the constraints can always be numbered in a way such that* $\{\mathbf{R_1} \ldots \mathbf{R_{n-1}}\}$ *is the set* $\{\mathbf{R_I}\}$

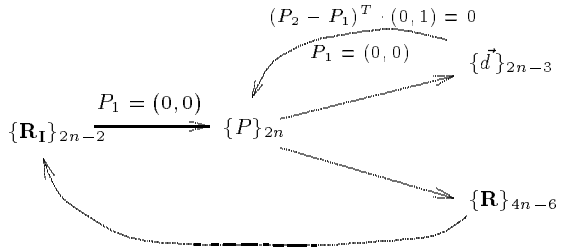

Figure 2: Relations between the different space representation

Proof. Let first observe that a subset of oriented constraints $\{\mathbf{R_1} \ldots \mathbf{R_{n-1}}\}$ is a set $\{\mathbf{R_I}\}$ iff every point $\mathbf{P}_k$ in the model can be reached from $\mathbf{P}_1$ through a path only involving arcs corresponding to constraints in the subset. In this case, every element of $\{\mathbf{P}\}$ can be computed as a linear combination of elements of $\{\mathbf{R_I}\}$. Of course, the lemma is true for $n = 2$ and $m = 1$. Let assume that it is true for n points. In this case, any of the $\mathbf{P}_k, k = 1 \ldots n$ points can be reached from $\mathbf{P}_1$ through a path involving only oriented constraints in $\{\mathbf{R_1} \ldots \mathbf{R_{n-1}}\}$. For $n+1$, the lemma remains true provided that the new element R_n to be added to $\{\mathbf{R_I}\}$ is a constraint between the new point $\mathbf{P}_{n+1}$ and any of the previous n points $\mathbf{P}_1 \ldots \mathbf{P}_n$. This is obviously always possible. □

As a conclusion, $\{\mathbf{P}\}$ can be computed from $\{\mathbf{R_I}\}$. On the other hand, the sets $\{d\}$ and $\{\mathbf{R}\}$ can be computed from $\{\mathbf{P}\}$ at is shown in figure 2 (the set suffixes represent their degrees of freedom).

Solving the distance constraints and reaching a solution $\vec{d} = \vec{d}^c$ can be understood as an optimization in the space $\{\mathbf{R_I}\}$, which is a subspace of $\{\mathbf{R}\}$. The goal is to obtain a set of independent oriented constraints $\{\mathbf{R_1} \ldots \mathbf{R_{n-1}}\}$ such that $||\mathbf{R_k}||^2 = 1$ for every $\mathbf{R_k} \in \{\mathbf{R}\}$. In other words and defining

$$\triangle_k = ||\mathbf{R}_k||^2 - 1 = (d_k/d_k^c)^2 - 1$$

that can also be expressed,

$$\triangle_k = \frac{{d_k}^2}{d_k^c} - 1 = \frac{(d_k + d_k^c)(d_k - d_k^c)}{(d_k^c)^2} = \frac{d_k + d_k^c}{(d_k^c)^2}\delta_k$$

A solution $\vec{d} = \vec{d}^c$ has been reached iff $\triangle_k = 0 \; \forall \mathbf{R_k} \in \{\mathbf{R}\}$. By definition of $\triangle_k$,

$$\begin{aligned} ||R_k|| = 1 \quad &\Leftrightarrow \triangle_k = 0 \quad \Leftrightarrow \delta_k = d_k - d_k^c = 0 \\ ||R_k|| > 1 \quad &\Leftrightarrow \triangle_k > 0 \quad \Leftrightarrow \delta_k > 0 \\ ||R_k|| < 1 \quad &\Leftrightarrow \triangle_k < 0 \quad \Leftrightarrow \delta_k < 0 \end{aligned}$$

The geometric relaxation solver works by differentially modifying the independent oriented constraints $\mathbf{R_k} \in \{\mathbf{R_I}\}, \; k = 1 \ldots n-1$ at each iteration. The algorithm is based on the following lemma,

Lemma 5.2. *Let assume that for every oriented constraint $\mathbf{R_k} \in \{\mathbf{R}\}$, a differential modification $d\mathbf{R_k}$ is performed such that*

$$\begin{aligned} &\textit{If} \quad ||\mathbf{R_k}|| < 1 \; , \quad \mathbf{R_k} \cdot d\mathbf{R_k} > 0 \\ &\textit{If} \quad ||\mathbf{R_k}|| = 1 \; , \quad \mathbf{R_k} \cdot d\mathbf{R_k} = 0 \\ &\textit{If} \quad ||\mathbf{R_k}|| > 1 \; , \quad \mathbf{R_k} \cdot d\mathbf{R_k} < 0 \end{aligned}$$

Then, in the case the total energy E_T of the model is 0 it remains unchanged. On the other hand, if the total energy E_T is positive, it decreases.

Proof. The proof of the lemma is straightforward, and it is based on writing the total energy E_T of the system in terms of the deviations of $\triangle_k$.

In the case $E_T = 0$, taking into account that $E_T = \sum_i \delta_i^2$, it follows that $\delta_i = 0 \; \forall i$. Then, $\triangle_i = 0 \; \forall i$ and $||R_i|| = 1 \; \forall i$. But, $d(\triangle_i) = d(\mathbf{R_i}\mathbf{R_i} - 1) = 2\mathbf{R_i}d\mathbf{R_i} = 0$ from the hypothesis of the lemma. Now, $d(\triangle_i) \; \forall i$ implies that every $\triangle_i = 0$ remains unchanged. Consequently, $\delta_i = 0 \; \forall i$ and $E_T = 0$ after the differential modification.

In the case $E_T \neq 0$,

$$\begin{aligned} d(E_T) \quad &= 2\textstyle\sum_i \delta_i d(\delta_i) \quad = \\ &= 2\textstyle\sum_i \frac{(d_i^c)^2}{d_i + d_i^c} \triangle_i \frac{(d_i^c)^2}{2d_i} d(\triangle_i) \quad = \\ &= 2\textstyle\sum_i \frac{(d_i^c)^4}{2d_i(d_i + d_i^c)} \triangle_i d(\triangle_i) \end{aligned}$$

The first term of the sumatory is always positive and given that

$$d(\triangle_i) = d(\mathbf{R_i}\mathbf{R_i} - 1) = 2\mathbf{R_i}d(\mathbf{R_i})$$

It is possible to say that all not null terms of $d(E_T)$ are negatives since, by hypothesis, $\triangle_i$ has an oppositte sign of $d(\triangle_i) \; \forall i$. Therefore,

$$d(E_T) < 0$$

□

Now, the following lemma guarantees that it is always possible to compute a set of differential modifications $d\mathbf{R_1} \ldots d\mathbf{R_{n-1}}$ such that the lemma 5.2 is fulfilled.

Lemma 5.3. *In a well-constrained model, given a set of independent oriented constraints* $\mathbf{R_I} \in \{\mathbf{R_1} \ldots \mathbf{R_{n-1}}\}$*, it is always possible to compute a set of differential modifications* $d\mathbf{R_1} \ldots d\mathbf{R_{n-1}}$ *such that lemma 5.2 is fulfilled.*

Proof. The proof is based on the analysis of the linear system derived from lemma 5.2:

$$[\mathbf{A}] \cdot [d\mathbf{R}] = [\mathbf{b}]$$

Each equation in this system expresses one of the differentials $\mathbf{R_k} d(\mathbf{R_k}),\ \mathbf{R_k} \in \{\mathbf{R}\}$ as a function of the independent differential modifications $d(\mathbf{R_i}),\ \mathbf{R_i} \in \{\mathbf{R_I}\},\ \mathbf{R_i} = (x_i, y_i)$:

$$\begin{aligned} \mathbf{R_k} d(\mathbf{R_k}) = & \quad \sum_{i=1}^{n-1} \mathbf{A_{ki}} d(\mathbf{R_i}) \quad = \\ = & \quad \sum_{j=1}^{n-1} a_{kj} dx_j + \sum_{j=n}^{2n-2} a_{kj} dy_{j-(n-1)} \end{aligned}$$

In this system $[\mathbf{A}]$ is a $(2n-3) \times (2n-2)$ matrix, $[d\mathbf{R}]$ is the $(2n-2)$ vector with the unknown values of the differential modifications $d\mathbf{R_1} \ldots d\mathbf{R_{n-1}}$ of the independent oriented constraints, and $[\mathbf{b}]$ is the vector with the inner products, whose signs depend on $||\mathbf{R_k}||$, as required by lemma 5.2. Each equation $-$ row in $[\mathbf{A}]-$ represents the inner product $\mathbf{R_k} \cdot d\mathbf{R_k}$ for every oriented constraint $\mathbf{R_k} \in \{\mathbf{R}\},\ k = 1 \ldots 2n-3$, having written $d\mathbf{R_k}$ as the corresponding combination of the independent differential modifications $d\mathbf{R_i},\ i = 1 \ldots n-1$ (see example in section 5.2). The matrix $[\mathbf{A}]$ therefore depends on the location of the points in $\{\mathbf{P}\}$. It is constant for a given iteration, but varies from one iteration to the next one.

Coming back to the linear system $[\mathbf{A}] \cdot [d\mathbf{R}] = [\mathbf{b}]$ it turns out that (provided that the system is well-constrained) the dimension of the null space of $[\mathbf{A}]$ is 1. The whole system has one degree of freedom once the point $\mathbf{P}_1$ has been fixed $-$ rigid rotations around $\mathbf{P}_1$ $-$. Any of the components of the vector $d\mathbf{R}$ can be assigned an arbitrary value and the rigid motion of the system around P_1 will determine the others. But, the dimension of the null space of $\mathbf{A}$ cannot be greather than 1, as it is not possible to assign arbitrary values to more than one scalar component of $d\mathbf{R}$. This would produce non-rigid deformations of the system, which is not possible in a well-constrained system. □

As a consequence, the linear system $[\mathbf{A}] \cdot [d\mathbf{R}] = [\mathbf{b}]$ has always a solution and the set $\{\mathbf{R_I}\}$ can always be differentially modified in a way that the total energy E_T decreases and the norms $||\mathbf{R_k}||$ approach to the unity. Therefore, the following proposition is derived.

Proposition 5.1. *An iterative geometric relaxation solver based on the differential modifications of the independent oriented constraints computed by solving the linear system* $[\mathbf{A}] \cdot [d\mathbf{R}] = [\mathbf{b}]$ *converges to the global minimum with* $E_T = 0$.

As a final observation, the solution of the linear system can be used in a non differential way. Assigning,

$$\mathbf{R_i} := \mathbf{R_i} + \alpha \cdot d\mathbf{R_i},\ i = 1 \ldots n-1$$

the relaxation parameter α can be tuned in order to optimize the energy decreasing at each iteration.

5.2 Example of the linear system

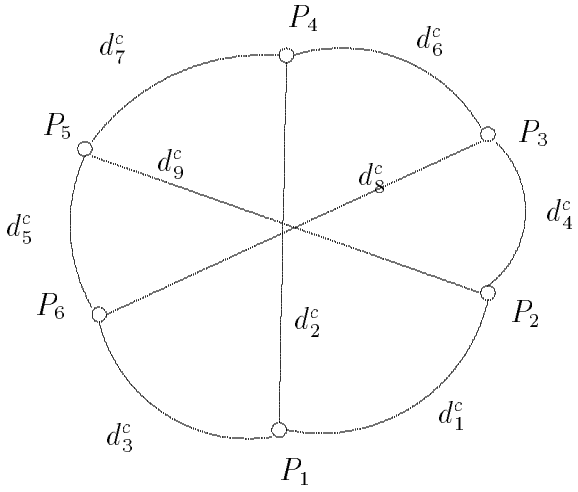

Figure 3: Example of 6 points well-constrained with 9 distance constraints

Points:

$$\{\mathbf{P}\} = \{\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4, \mathbf{P}_5, \mathbf{P}_6, \}$$

Constraints:

$$\begin{aligned}
d_1^c &= C(\mathbf{P}_1, \mathbf{P}_2) \\
d_2^c &= C(\mathbf{P}_1, \mathbf{P}_4) \\
d_3^c &= C(\mathbf{P}_1, \mathbf{P}_6) \\
d_4^c &= C(\mathbf{P}_2, \mathbf{P}_3) \\
d_5^c &= C(\mathbf{P}_5, \mathbf{P}_6) \\
d_6^c &= C(\mathbf{P}_3, \mathbf{P}_4) \\
d_7^c &= C(\mathbf{P}_4, \mathbf{P}_5) \\
d_8^c &= C(\mathbf{P}_3, \mathbf{P}_6) \\
d_9^c &= C(\mathbf{P}_2, \mathbf{P}_5)
\end{aligned}$$

Oriented constraints:

$$\begin{aligned}
\mathbf{R}_1 &= (\mathbf{P}_2 - \mathbf{P}_1)/d_1^c \\
\mathbf{R}_2 &= (\mathbf{P}_4 - \mathbf{P}_1)/d_2^c \\
\mathbf{R}_3 &= (\mathbf{P}_6 - \mathbf{P}_1)/d_3^c \\
\mathbf{R}_4 &= (\mathbf{P}_3 - \mathbf{P}_2)/d_4^c \\
\mathbf{R}_5 &= (\mathbf{P}_5 - \mathbf{P}_6)/d_5^c
\end{aligned}$$

$$\mathbf{R}_6 = (\mathbf{P}_4 - \mathbf{P}_3)/d_6^c$$
$$\mathbf{R}_7 = (\mathbf{P}_5 - \mathbf{P}_4)/d_7^c$$
$$\mathbf{R}_8 = (\mathbf{P}_6 - \mathbf{P}_3)/d_8^c$$
$$\mathbf{R}_9 = (\mathbf{P}_5 - \mathbf{P}_2)/d_9^c$$

$$\{\mathbf{R}\} = \{\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3, \mathbf{R}_4, \mathbf{R}_5, \mathbf{R}_6, \mathbf{R}_7, \mathbf{R}_8, \mathbf{R}_9\}$$

$$\{\mathbf{R_I}\} = \{\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3, \mathbf{R}_4, \mathbf{R}_5\}$$

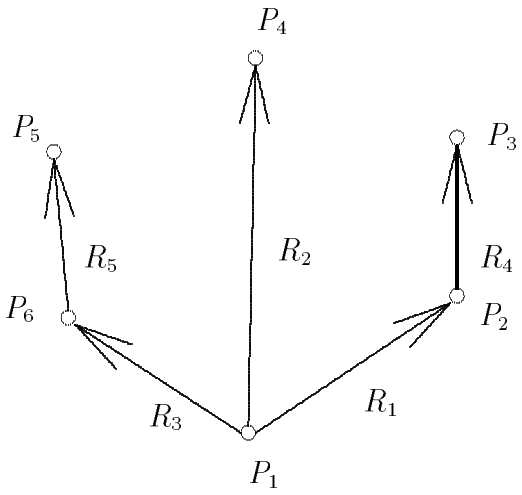

Figure 4: The set of $\{\mathbf{R_I}\}$ oriented constraints

$\{\mathbf{R_I}\}$ to $\{\mathbf{P}\}$ conversion

$$\begin{aligned}
\mathbf{P}_1 &= (0,0) \\
\mathbf{P}_2 &= \mathbf{P}_1 + d_1^c\mathbf{R}_1 = d_1^c\mathbf{R}_1 \\
\mathbf{P}_4 &= d_2^c\mathbf{R}_2 \\
\mathbf{P}_6 &= d_3^c\mathbf{R}_3 \\
\mathbf{P}_3 &= \mathbf{P}_2 + d_4^c\mathbf{R}_4 = d_1^c\mathbf{R}_1 + d_4^c\mathbf{R}_4 \\
\mathbf{P}_5 &= \mathbf{P}_6 + d_5^c\mathbf{R}_5 = d_3^c\mathbf{R}_3 + d_5^c\mathbf{R}_5
\end{aligned}$$

$\{\mathbf{P}\}$ to $\{\mathbf{R}\}$ conversion

$$\begin{aligned}
\mathbf{R}_1 &= (\mathbf{P}_2 - \mathbf{P}_1)/d_1^c \\
\mathbf{R}_2 &= (\mathbf{P}_4 - \mathbf{P}_1)/d_2^c \\
\mathbf{R}_3 &= (\mathbf{P}_6 - \mathbf{P}_1)/d_3^c
\end{aligned}$$

$$
\begin{aligned}
\mathbf{R}_4 &= (\mathbf{P}_3 - \mathbf{P}_2)/d_4^c \\
\mathbf{R}_5 &= (\mathbf{P}_5 - \mathbf{P}_6)/d_5^c \\
\mathbf{R}_6 &= (d_2^c\mathbf{R}_2 - d_1^c\mathbf{R_1} - d_4^c\mathbf{R_4})/d_6^c \\
\mathbf{R}_7 &= (d_3^c\mathbf{R}_3 + d_5^c\mathbf{R}_5 - d_2^c\mathbf{R}_2)/d_7^c \\
\mathbf{R}_8 &= (d_3^c\mathbf{R}_3 - d_1^c\mathbf{R}_1 - d_4^c\mathbf{R}_4)/d_8^c \\
\mathbf{R}_9 &= (d_3^c\mathbf{R}_3 + d_5^c\mathbf{R}_5 - d_1^c\mathbf{R}_1)/d_9^c
\end{aligned}
$$

In the example of the figure 3 a set of possible values are $d_i = 1$ and $d_i^c = 2$ for $i = 1,3,4,5,6,7$ whereas $d_i = 2$ and $d_i^c = 1$ for $i = 2,8,9$. Then, if $\mathbf{R}_i$ is represented by $\mathbf{R}_i = (x_i, y_i)$

In the first iteration,

$$
\begin{aligned}
x_i dx_i + y_i dy_i &> 0 \quad \text{in } \mathbf{R_1}, \mathbf{R_3}, \mathbf{R_4}, \mathbf{R_5}, \mathbf{R_6}, \mathbf{R_7}. \\
x_i dx_i + y_i dy_i &< 0 \quad \text{in } \mathbf{R_2}, \mathbf{R_8}, \mathbf{R_9}.
\end{aligned}
$$

At each iteration it is necessary to calculate the sign for each $\mathbf{R}_i$ and the linear system to solve will be,

$$[\mathbf{A}] \cdot [d\mathbf{R}] = \mathbf{b}, \text{ with } b_i > 0 \ \forall i$$

$$
\begin{pmatrix}
x_1 & 0 & 0 & 0 & 0 & y_1 & 0 & 0 & 0 & 0 \\
0 & x_2 & 0 & 0 & 0 & 0 & y_2 & 0 & 0 & 0 \\
0 & 0 & x_3 & 0 & 0 & 0 & 0 & y_3 & 0 & 0 \\
0 & 0 & 0 & x_4 & 0 & 0 & 0 & 0 & y_4 & 0 \\
0 & 0 & 0 & 0 & x_5 & 0 & 0 & 0 & 0 & y_5 \\
\frac{-d_1^c x_6}{d_6^c} & \frac{d_2^c x_6}{d_6^c} & 0 & \frac{-d_4^c x_6}{d_6^c} & 0 & \frac{-d_1^c y_6}{d_6^c} & \frac{d_2^c y_6}{d_6^c} & 0 & \frac{-d_4^c y_6}{d_6^c} & 0 \\
0 & \frac{-d_2^c x_7}{d_7^c} & \frac{d_3^c x_7}{d_7^c} & 0 & \frac{d_5^c x_7}{d_7^c} & 0 & \frac{-d_2^c y_7}{d_7^c} & \frac{d_3^c y_7}{d_7^c} & 0 & \frac{d_5^c y_7}{d_7^c} \\
\frac{-d_1^c x_8}{d_8^c} & 0 & \frac{d_3^c x_8}{d_8^c} & \frac{-d_4^c x_8}{d_8^c} & 0 & \frac{-d_1^c y_8}{d_8^c} & 0 & \frac{d_3^c y_8}{d_8^c} & \frac{-d_4^c y_8}{d_8^c} & 0 \\
\frac{-d_1^c x_9}{d_9^c} & 0 & \frac{d_3^c x_9}{d_9^c} & 0 & \frac{d_5^c x_9}{d_9^c} & \frac{-d_1^c y_9}{d_9^c} & 0 & \frac{d_3^c y_9}{d_9^c} & 0 & \frac{d_5^c y_9}{d_9^c}
\end{pmatrix}
\begin{pmatrix} dx_1 \\ dx_2 \\ dx_3 \\ dx_4 \\ dx_5 \\ dy_1 \\ dy_2 \\ dy_3 \\ dy_4 \\ dy_5 \end{pmatrix}
=
\begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \\ b_{10} \end{pmatrix}
$$

In this system, negative expressions $\mathbf{R_k} d\mathbf{R_k} < 0$ must be multiplied by (-1), in order to have only positive values in the independent vector $\mathbf{b}$.

6 Model Conversion to $d - reduced$ form

The models supported by the IMR only deal with distance constraints between two points, angle constraints between two vectors with coincident end points and angle constraints between two vectors with non-coincident end points. Models having only distance constraints between two points and angle constraints between two vectors with coincident end points will be called $3P - angle$ models. Models having distance constraints and angle

constraints between two vectors with non-coincident end points will be called $4P-angle$ models.

Definition 6.1. A model $\mathcal{M}^{'}$ is *d-reduced* if it has only distance constraints between points.

Definition 6.2. A well-constrained model $\mathcal{M}$ with angle and distance constraints is *d-reducible* if it is possible to find a *d-reduced* model $\mathcal{M}^{'}$ such that every solution of $\mathcal{M}^{'}$ is a solution of the initial model $\mathcal{M}$ and viceversa. A solution is a set of point locations that guarantees $\vec{d}=\vec{d}^{c}$.

In the figure 5 the model $\mathcal{M}$ is equivalent to the *d-reduced* model $\mathcal{M}^{'}$. In order to solve the initial model $\mathcal{M}$ the first step is to convert it to $\mathcal{M}^{'}$ and compute the values of new distance constraints in terms of the angle constraint values. In a second step, the iterative relaxation solver can be used to achieve a solution for $\mathcal{M}^{'}$. Finally, the point locations in $\mathcal{M}^{'}$ give a solution for $\mathcal{M}$.

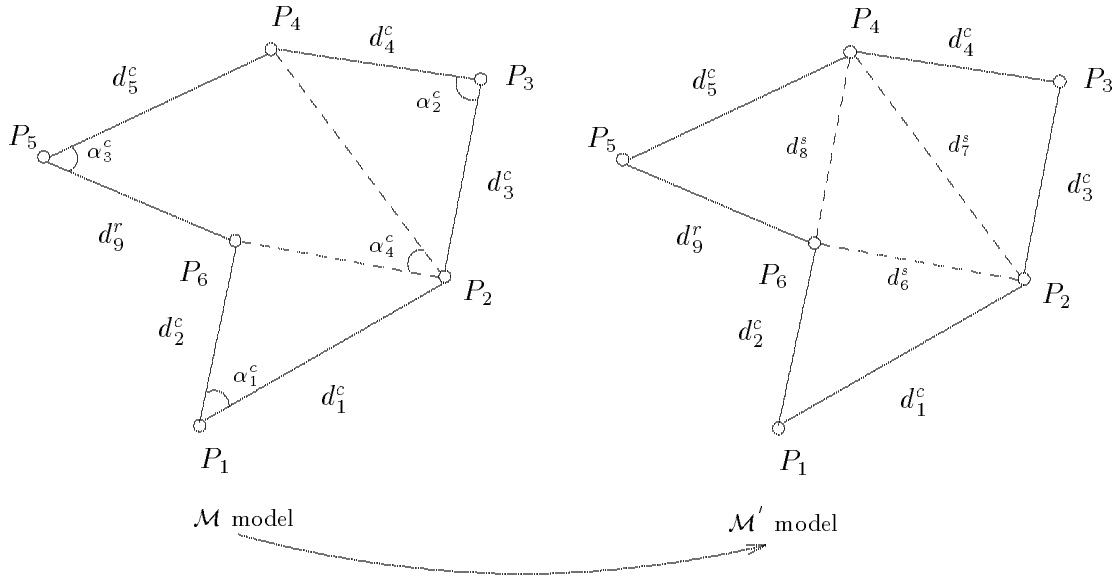

Figure 5: Equivalence between 2 models

6.1 $3P-angle$ Model Conversion to $d-reduced$ form

In this section we work on 2D models with distance constraints between two points and angle constraints between two vectors with coincident end point. Angle constraints involve three points. The main goal is to find a $d-reduced$ model $\mathcal{M}^{'}$ with only distance constraints such that every solution of $\mathcal{M}^{'}$ is a solution of the initial model $\mathcal{M}$ and vice-versa.

Related with distance d_i and angle α_i constraints, the following definitions can be stated:

Definition 6.3. The three points involved in an angle constraint between two vectors with a common endpoint form a triangle called *reduction triangle*

Definition 6.4. The *reduction distances* d^r of an angle constraint are each one of the side distances of the corresponding reduction triangle.

Definition 6.5. A reduction distance is a *substitution distance* d^s when there doesn't exist any distance constraint d^c defined on it in the set of constraints of the model.

Property 6.1. *If a model $\mathcal{M}$ is well-constrained, in every reduction triangle there exists at least one free reduction distance d^r that can be a substitution distance d^s.*

Proof. A well-constrained triangle can have only three distance or angle constraints of the total set of constraints fixed by the model. If in the reduction triangle there isn't any free reduction distance, the triangle has four constraints and it doesn't fulfil the well-constrained condition fixed by Lambert (2.2). □

Definition 6.6. Given a well-constrained model $\mathcal{M}$ with distance and angle constraints a *simple d-reduction* is the process of changing a single angle constraint α_i by a *substitution distance* constraint d_i^s of its associated reduction triangle. The *substitution distance* is one of the *reduction distances* of the triangle so that after the *simple d-reduction* every point in the reduction triangle has at least 2 constraints defined on it, in order to mantain the model well-constrained.

Definition 6.7. The *d-reduction process* is the iterative application of *simple d-reductions* until a *d-reduced* model is achieved.

Property 6.2. *At each simple d-reduction the number of angle constraints r_a is decreased in one, the number of distance constraints r_d is increased in one and the total constraints remains the same and the model remains well-constrained*

The model remains well-constrained because at least there are 2 constraints on every point of the triangle reduction. If the number of constraints is the same at each *simple d-reduction* wether the model is over-constrained, it means that exists some point with only one constraint on it.

Proposition 6.1. *In a d-reduction process, the application of a simple d-reduction on a $3P-angle$ model $\mathcal{M}$ produces an equivalent model $\mathcal{M}'$ that has the same solution than $\mathcal{M}$.*

Proof. This is equivalent to prove that for any angle constraint α_i it be can find a substitution distance d_i^s such that

$$\forall\ \alpha_j \neq \alpha_i\ \exists\ d_j^s \mid d\_reduction(\alpha_j) = d_j^s \text{ and } d_i^s \neq d_j^s$$

In a reduction triangle the following cases may appear:

$$\begin{aligned}
M(\alpha_1, d_1^c, d_2^c, d_1^r) &= M'(d_1^c, d_2^c, d_1^s(\alpha_1)) \\
M(\alpha_1, d_1^c, d_1^r, d_2^r) &= M'(d_1^c, d_1^s(\alpha_1), d_2^r) \\
M(\alpha_1, d_1^c, d_1^r, d_2^r) &= M'(d_1^c, d_1^r, d_2^s(\alpha_1)) \\
M(\alpha_1, d_1^r, d_2^r, d_3^r) &= M'(d_1^s(\alpha_1), d_2^r, d_3^r) \\
M(\alpha_1, d_1^r, d_2^r, d_3^r) &= M'(d_1^r, d_2^s(\alpha_1), d_3^r) \\
M(\alpha_1, d_1^r, d_2^r, d_3^r) &= M'(d_1^r, d_2^r, d_3^s(\alpha_1))
\end{aligned}$$

In all the cases, it is immediate to check that both models are equivalent. Therefore, the application of a *simple d-reduction* always produces an equivalent model with the same solution than the initial one. □

Proposition 6.2. *In a well-constrained model for each reduction triangle there always exists at least one of its reduction distances that is not involved in other simple d-reductions.*

Proof. In a given reduction step, if there is a reduction triangle where all the *reduction distances* are constraints defined in the model d^r or *substitution distances* d^s of other triangles, the model is over-constrained because the triangle is over-constrained with 4 constraints over 3 points and it doesn't fulfil the Lambert property. However, this over-constrained model has been obtained from the initial model through a sequence of *simple d-reductions* that produce equivalent models. As a consequence, the initial model must be over-constrained. □

6.2 $4P - angle$ Model Conversion to $d - reduced$ form

In this section the *d-reduction* process in 2D models with distance constraints between two points and angle constraints between two vectors with no coincident end point is analyzed. In angle constraints there are involved four points are involved (figure 6).

Let state the following definitions:

Definition 6.8. The four points involved in a angle constraint between two vectors with a non common endpoint form a quadrilateral polygon called *reduction quadrilateral*

Definition 6.9. The *reduction distances* d^r of an angle constraint are each side distances and diagonal distances of the corresponding reduction quadrilateral.

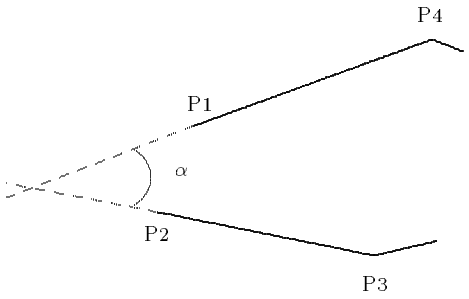

Figure 6: Angle constraint involving 4 points

On the basis of these definitions it is possible to extend the properties of the $3P - angle$, exposed in the last section, to the $4P - angle$ Now, instead of reduction triangle it will be talked about reduction quadrilateral. *Simple d-reduction* can also be applied in the case of $4P - angle$ models.

Property 6.3. *If a model $\mathcal{M}$ is well-constrained, in every reduction quadrilateral there exists at least two free reduction distances d^r that can be substitution distances d^s.*

Proof. In a quadrilateral there are 6 possible *reduction distances* between the 4 involved points P_1, P_2, P_3, P_4, these are $d_{12}, d_{23}, d_{34}, d_{14}, d_{13}, d_{24}$. With 4 points with an angle constraint among them, 4 distance constraints from the 6 possible ones are needed . Therefore, at least 2 of them don't have defined constraints and they can be *substitution distance.* □

Property 6.4. *In a d-reduction process, the application of a simple d-reduction on a $4P - angle$ model $\mathcal{M}$ produces an equivalent model $\mathcal{M}'$ that has the same solution than $\mathcal{M}$.*

Proof. Figure 7 shows the possible cases of *reduction distances* that can appear *substitution distance* in an angle involving 4 points. In all cases, it is immediate to check that the application of a *simple d-reduction* always produces an equivalent model with the same solution than the initial one.

In the figure it is possible to observe that the d_2 and d_4 distance constraints can also be *substitution distance.* In $4P - angle$ models there are primary substitution distances as d_1, d_3, d_5, d_6. If these primary substitution distances are free and they are not involved in other substitutions, they should be used instead of the secondary substitution distances d_2 or d_4.

□

Property 6.5. *In a well-constrained model, for each reduction quadrilateral there always exists at least one of its reduction distances that is not involved in any other simple d-reduction.*

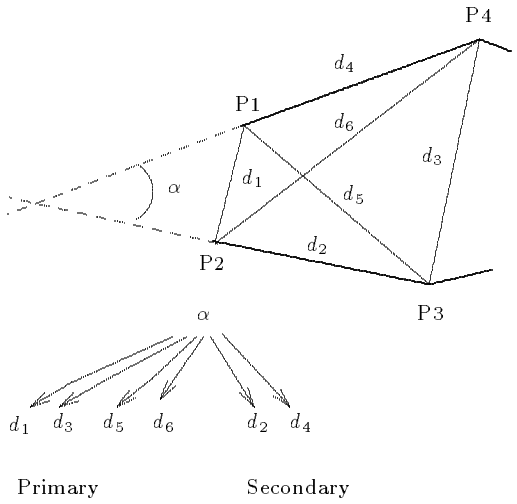

Figure 7: *Simple d-reduction* posibilities in an angle on a $4P - angle$ model

Proof. In a given reduction step, if there is a reduction quadrilateral where all the *reduction distances* are constraints defined in the model d_c or *substitution distances* d^s of other reduction element, the model is over-constrained because the quadrilateral is over-constrained with 6 constraints over 4 points and it doesn't fulfil the Lambert property. But this over-constrained model has been obtained from the initial model through a sequence of *simple d-reductions* that produce equivalent models. As a consequence, the initial model must be over-constrained. □

Theorem 6.1. *In a well-constrained model with angle and distance constraints it is always possible to obtain a d-reduced model such that after a d-reduction process all the angle constraints have been expressed as distance constraints.*

Proof. From the set of properties seen before. □

Corolarium 6.1. The convergence demostrations of the proposed relaxation method in order to achieve constraint satisfaction for *d-reduced* models is valid for *d-reducible* models.

6.3 The *simple d-reduction* process

As it has been shown above, the *d-reduction* process is based on the iterative substitution of every angle constraint by one of its *substitution distance*, such that each angle constraint has its corresponding different *substitution distance*. The scheme to follow in order to have the constraint satisfaction is:

1. Perform the *d-reduction* process if angle constraints exist in the model.

2. Satisfy by relaxation the *d-reduced* model. At each iteration angle and *substitution distance* values are calculated.

The *d-reduction* is performed in the following way:

Given a set of angle and the set of avalaible *reduction distances*. Due to these two set are disjoints the relation between them can be represented by a bipartite graph (bigraph). The *d-reduction* process is a *perfect matching* problem between the set of angle nodes and the set of distance nodes. A *perfect matching* where each angle α_i has one *reduction distance* d^r which is its *substitution distance* $d^s(\alpha_i)$. As a result the *perfect matching* is generated a set of pairs $< \alpha_i, d_k^r >$ corresponding to the *simple d-reductions* necessaries for the *d-reduction* process. In the *perfect matching* a *reduction distance* of each angle is selected to be its *substitution distance*

The *perfect matching* problem can be solved using the algorithm proposed in [AHU83] in order to find maximal paths in a bigraph. However, it is possible to reduced the search space using a preprocess based on the following properties:

Property 6.6. *All angle node α_i which in the bigraph has only one arc to a distance node d_i^r and the arcs involved can be erased from the bigraf because the matching (substitution) of $< \alpha_i >$ and $< d_k^r >$ it is necessary to achieve the perfect matching state.*

Property 6.7. *All distance node d_i^r which in the bigraph has only one arc to a distance node α_i and the arcs involved can be erased from the bigraf because the matching (substitution) of $< \alpha_i >$ by $< d_k^r >$ it is necessary to achieve the perfect matching state.*

The application of the above properties as a prepocess can solve totally the *perfect matching* or it can resulted a bigraf where the degree of the angle nodes is 2. In this case it is easy to achieve the perfect matching state without using the algorithm proposed by [AHU83].

7 Single Point, Single Constraint and Hybrid Relaxation Solvers

For $d-reduced$ systems a constraint solver is next proposed based on an iterative relaxation method. In this way, while $||\vec{\delta}|| \geq \varepsilon$, at each iteration, the constraint solver works over points or over constraints. This means that the solver in each step has to decide to select one constraint to relax or to perform a global deformation in order to move slightly all the points. We are not using this case.

Up to now the following cases have been adressed:

- One-point relaxation: On a point P, if the $grad(\vec{\delta}) \neq 0$ is no null, P can be moved in the direction of the gradient vector. In this way the maximum decrease of $|\vec{\delta}||$ is achieved. The problem raises when there is a local minimum and the gradient vector is null for all points.

- One-constraint relaxation: It is also possible to relax one constraint d_i. In this case local minima can be reached. If the constraint relaxed d_i has the maximum value of $|\vec{\delta}|$ a good behavior of the convergence of the solver has been experimentally checked.
- Global point relaxation: In this case a space deformation is applied in order to preserve the decreasing of $||\vec{\delta}||$.
- Global constraint relaxation: This case would require to solve all the constraints at the same time. Therefore, it is unaffordable.

Constraint Variation Matrix $A = [a_{ji}]$: each matrix element represents the differential variation of the parameter associated with constraint j according to the variation of the constraint i ($i \neq j$). So, a_{ji} represents, the differential variation of the parameter d_j according to the variation of d_i.

$$dim(A) = m \cdot m$$

$$A = \begin{cases} a_{ji} = 1 & \text{if } i = j \\ a_{ji} = 0 & \text{if } i \neq j \text{ and any change in } d_i \text{ doesn't produces a variation on } d_j \\ a_{ji} = \frac{\partial d_j}{\partial d_i} & \text{if } i \neq j \ d_i, d_j \text{ share a common endpoint} \end{cases}$$

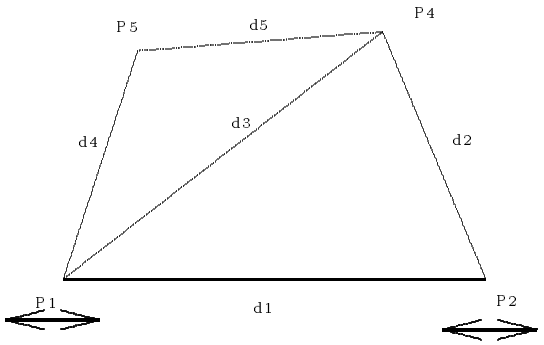

Figure 8: Link between distance constraints

The matrix column k represents the unit variation of the overall constraints that are linked to d_k. Because the solver works with distance constraints the constraint variation matrix is symmetric and it is easy to prove that $|a_{ji}| \leq 1$.

Using the one-constraint relaxation, it can be shown [SoB98] that the solver converges to the solution if the constraint δ_i is so that

$$\delta_i = \max_j |\delta_j|$$

and there exists a positive value L such that,

$$\sum_j (\delta_\kappa)_j^* \, b_{ji} \;\geq\; L \;>\; 0 \tag{1}$$

with

$$\begin{cases} b_{ji} = \frac{a_{ji}}{\sqrt{\sum_j a_{ji}^2}} \\ (\delta_\kappa)_j^* = \frac{(\delta_\kappa)_j}{(\delta_\kappa)_i} \end{cases}$$

The equation 1 can be understood as a global concordance requirement, because the relative signs of $(\delta_\kappa)_j$ with respect to $(\delta_\kappa)_i$, must agree with the a_{ji} signs. The equation 1 states that the one-constraint relaxation algorithm converges in global concordance cases. If 1 is not satisfied the relaxation can fall into local minima and convergence can not be ensured (Anyway, we have experimentally detected a good convergence behaviour also in these cases).

As a consequence, a hybrid iterative constraint solver has been developed. In it, the relaxation process is performed in the following way,

Hybrid Constraint relaxation. At each iteration a constraint is relaxed or the whole set of points are deformed. The following algorithm is used:

If the system is not in a local minimum status then

- Choose a constraint i.
- Relax the constraint i ($d_i\rfloor_\kappa$) by making it closer to the equilibrium value d_i^e. A relaxation factor r is used.

$$d_i\rfloor_{\kappa+1} = d_i\rfloor_\kappa + r \cdot (d_i^e - d_i\rfloor_\kappa)$$

- Update the constraints d_j such that $a_{ji} \neq 0$

$$d_j\rfloor_{\kappa+1} = d_j\rfloor_\kappa + r \cdot a_{ji} \cdot (d_i^e - d_i\rfloor_{\kappa+1})$$

- Update the geometric element values (vectors, points, plane equations,..) affected by the modification of the constraint values.
- Compute the new matrix values of A and $\vec{\delta}$.

Else

- Use a global deformation in order to move the whole set of points and decrease $||\vec{\delta}||$.

8 Examples and discussion

In this section a set of six points that initially form a regular hexagon is studied. Between the points, the following distance constraints have been defined:

$$
\begin{aligned}
d_1^c &= distance(P_1P_2)\\
d_2^c &= distance(P_1P_4)\\
d_3^c &= distance(P_1P_6)\\
d_4^c &= distance(P_2P_3)\\
d_5^c &= distance(P_5P_6)\\
d_6^c &= distance(P_3P_4)\\
d_7^c &= distance(P_4P_5)\\
d_8^c &= distance(P_3P_6)\\
d_9^c &= distance(P_2P_5)
\end{aligned}
$$

In next sections it is shown the behavior of the proposed method with two set of specific values of the distance constraints using the proposed global deformation solver and the hybrid algorithm explained. In the second example and due to the constraint values used, the initial state is in a local minimum.

8.1 Example 1: L-shape

In this example, the nine distance constraints are fixed with the following values:

$$
\begin{aligned}
d_1^c &= 2\\
d_2^c &= 1.4142\\
d_3^c &= 2\\
d_4^c &= 1\\
d_5^c &= 1\\
d_6^c &= 1\\
d_7^c &= 1\\
d_8^c &= 2.2360\\
d_9^c &= 2.2360
\end{aligned}
$$

The figure 9 shows the initial configuration. On the other hand, the figure 10 presents the final configuration of the system.

In figure 11 the evolution of $||\vec{\delta}||$ at each iteration using the iterative global deformation solver is shown. In figure 12 the evolution of $||\vec{\delta}||$ at each iteration using the one-constraint

relaxation is presented. In figure 13 the evolution of $||\vec{\delta}||$ at each iteration using the proposed hybrid method is presented. In cases 11 and 13 the local minima are successfully avoided. Otherwise, local increases of $||\vec{\delta}||$ can be observed in case 12.

8.2 Example 2: Double triangular shape

In this example, the nine distance constraints are fixed with the following values:

$$\begin{aligned} d_1^c &= 2 \\ d_2^c &= 1 \\ d_3^c &= 2 \\ d_4^c &= 2 \\ d_5^c &= 2 \\ d_6^c &= 2 \\ d_7^c &= 2 \\ d_8^c &= 1 \\ d_9^c &= 1 \end{aligned}$$

The figure 14 and 15 show the initial and the final configuration of the system. In figures 16 and 18, the evolution of $||\vec{\delta}||$ at each iteration using the iterative global deformation solver and the hybrid method proposed are shown. In both cases the local minima are also successfully avoided using the proposed methods. Figure 17 show the evolution in one-constraint relaxation case.

9 3D Extension and Conclusions

A constraint solver that works on the Internal Model Representation and which is based only in distance constraints has been presented. Through an iterative relaxation of constraints a solution is found. The convergence of the solver using a global relaxation can be guaranteed.

The proposed method in 2D can be also used in 3D systems. The relaxation process starts from the initial conditions. In this way a mechanism is provided to reach a solution which is close to the initial conditions. In cases with several solutions, by changing the initial conditions it is possible to switch from one solution to the other. The solver can also be applied in under-constrained systems by keeping the user intention.

References

[Ald88] B.Aldfeld. Variation of geometries based on a geometric-reasoning method. CAD, vol.20, no.3, April 1988.

[AHO83] A.V.Aho, J.E.Hopcroft, J.D.Ullman. Data Structures and Algorithms. Addison-Wesley Publishing Co., 1983.

[Bor81] A.H.Borning. The programming language aspects of ThingLab, a constrained oriented simulation laboratory. ACM Trans. on Prog. Lang. and Systems, vol.3, no.4, October 1981.

[BFHCP93] W.Bouma, I.Fudos, C Hoffman, J.Cai, R.Paige. A geometric constraint solver. Technical Report, CSD-TR-93-054, Purdue University, 1993.

[Buch85] B.Buchberger. Gröbner Bases: An algorithmic Method in Polynomial Ideal Theory. In N.K.Bose, editor, Multidimensional Systems Theory, pp.184-232. D.Reidel PublishingCo., 1985.

[CFV88] U.Cugini, F.Folini, I.Vicini. A Procedural System for the Definition and Storage of Technical Drawings in Parametric Form. Proceedings of Eurographics'88, pp.183-196, North-Holland, 1988.

[Emm90] M.J.G.M.van Emmerick. Interactive Design of Parameterized 3D models by direct manipulation. PhD thesis, Delft University Press, 1990.

[FuH93] I.Fudos, C.Hoffman. Correctness proof of a geometric constraint solver. Technical Report CSD-93-076, Computer Sciences Departament, Purdue University, December 1993.

[HoJ92] C.M.Hoffmann, R.Juan. ERep. An editable high-level representation for geometric design and analysis. Technical Report CSD-TR-92-055. CAPO Report CER-92-24. Purdue Unversity, August 1992.

[HsB97] C.Hsu, B.D.Brüderlin. A Hybrid Constraint Solver Using Exact and Iterative Geometric Constructions. In CAD Systems Devolopment: Tools and Methods, D.Roller and P.Brunet Eds, Springer Verlag 1997, pp. 265-279

[JoS97] R. Joan-Arinyo, A. Soto. Rule-Based Geometric Constraint Solver. Computer & Graphics, vol.21 no.5, 1997.

[KaB90] D. Kalra, A.H. Barr. A Constraint-Based Figure-Maker. Proceedings of Eurographics'90, pp.413-424, North-Holland, 1990.

[Lam70] G.Laman. On Graphs and Rigidity of Plane Skeletal Structures. Journal of Engineering Mathematics, vol.4, no.4, October 1970.

[LGL81] V.C.Lin, D.C.Gossard, R.A.Light. Variational Geometry in Computer Aided Design. ACM Computer Graphics, vol.15, no.3, August 1981.

[Nel85] G.Nelson. Juno, a constraint-based graphics system. SIGGRAPH'85, vol.19, no.3, pp.235-243, San Francisco. July 22-26, 1985.

[Owe91] J.C.Owen. Algebraic Solution for Geometry from Dimensional Constraints. Proceedings of Symposium on Solid Modelling Foundations and CAD/CAM Applications. J.Rossignac, J.Turner (eds). Austin, June 5-7, pp.397-407, ACM Press 1991.

[RBN89] J.R.Rossignac, P.Borrel, L.R.Nackman. Interactive Design with Sequences of Parameterized Transformations. Intelligent CAD Systems II. V.Akman, P.J.W.ten Hagen, P.J. Veerkamp (eds), pp.93-125, Springer-Verlag, 1989.

[Rol91a] D.Roller. Advanced Methods for Parametric Design. Geometric Modelling. Methods and Applications. H.Hagen, D.Roller (eds), pp. 251-266, Springer-Verlag, 1991.

[Rol91b] D.Roller. An approach to computer-aided parametric design. CAD, vol.23, no.5, June 1991.

[SoB91c] W. Sohrt, B.D.Brüderlin. Interaction with constraints in 3D modeling. International Journal of Computational Geometry & Applications, vol.1, no.4, pp.405-425, 1991.

[SoB93] L.Solano, P.Brunet. A system for constructive constraint-based modelling. In B.Falcidieno and T.Kunii, editors, Modeling in Computer Graphics. Springer Verlag, 1993.

[SoB94a] L.Solano, P.Brunet. A language for constructive parametric solid modelling. Technical Report LSI-94-43-R, Universitat Politecnica de Catalunya, LiSI, 1994.

[SoB94b] L.Solano, P.Brunet. Constructive constraint-based model for parametric CAD systems. CAD, vol.26, no.8, pp.614-621, August 1994.

[SoB98] L.Solano, P.Brunet. Geometric Distance Constraint Satisfaction by Constraint-to-constraint Relaxation (Written in Catalan). Technical Report LSI-98-27-R, Universitat Politecnica de Catalunya, LiSI, 1998.

[StS80] G.L.Steele, G.L.Sussman. Constraint - a language for expressing almost-hierarchical descriptions. Artificial Intelligence, pp.1-39, January 1980.

[Sun88] G.Sunde. Specification of Shape by Dimensions and Other Geometric Constraints. Geometric Modelling for CAD Applications. M.J.Wozny, H.W.McLaughlin, J.L.Ecarnacao (eds), pp.199-213, North-holland 1988.

[Sut63] I.Sutherland. Sketchpad, a man-machine graphical communication system. In Proc. of the Spring Joint Comp. Conference, pp.329-345. IFIPS, 1963.

[VSR92] A.Verroust, F.Schonek, D.Roller. Rule-oriented method for parametrized computer aided design. CAD, vol.24, no.10, October 1992.

[Wen86] Wu Wen-Tsün. Basic principles of mechanical theorem proving in geometries. Journal of Systems Sciences and Mathematical Sciences, vol.4, pp.207-235, 1986.

[WFB87] A. Witkin, K.Fleischer, A. Barr. Energy Constraints on Parameterized models. Computer Graphics, vol.21, pp.225-232, 1987.

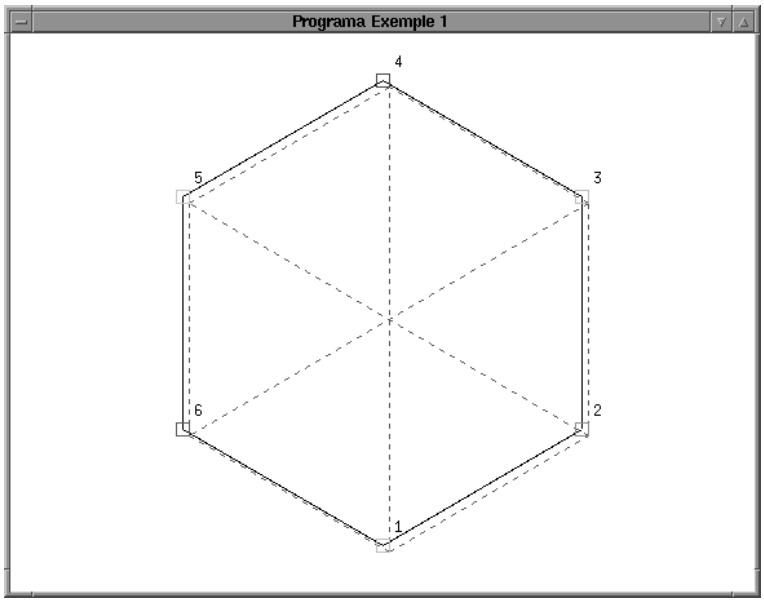

Figure 9: Initial hexagon configuration

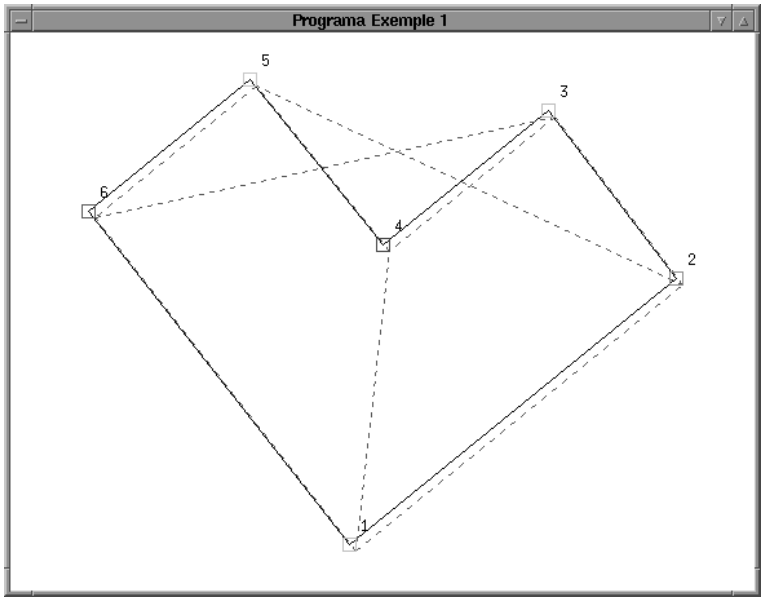

Figure 10: Final L-shape configuration

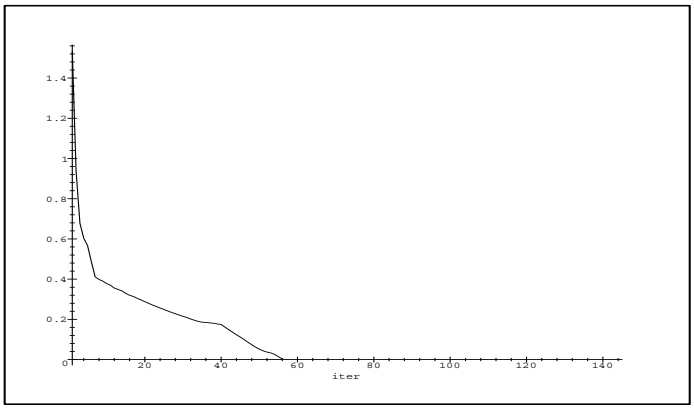

Figure 11: $||\vec{\delta}||$ evolution with global deformation

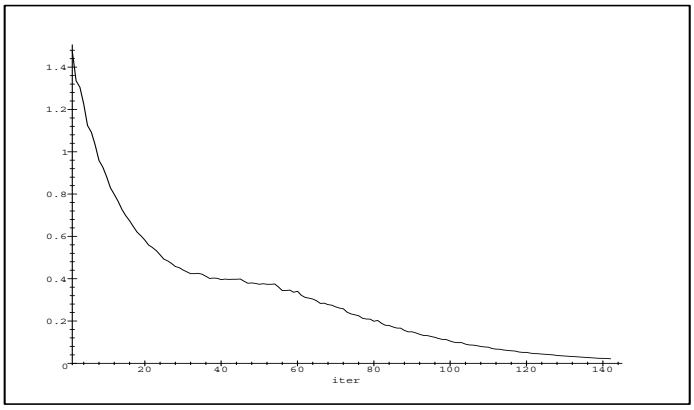

Figure 12: $||\vec{\delta}||$ evolution with one-contraint relaxation

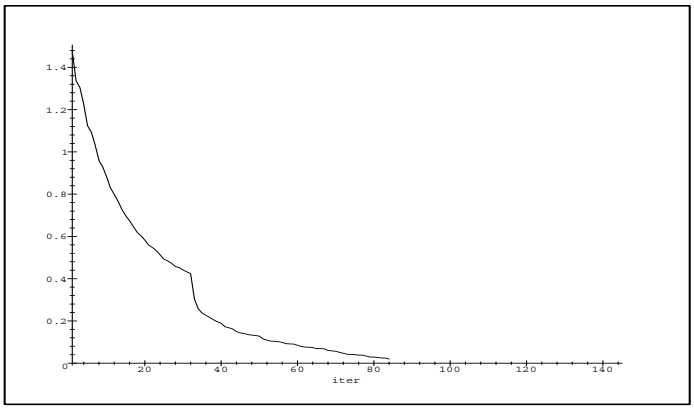

Figure 13: $||\vec{\delta}||$ evolution with hybrid method

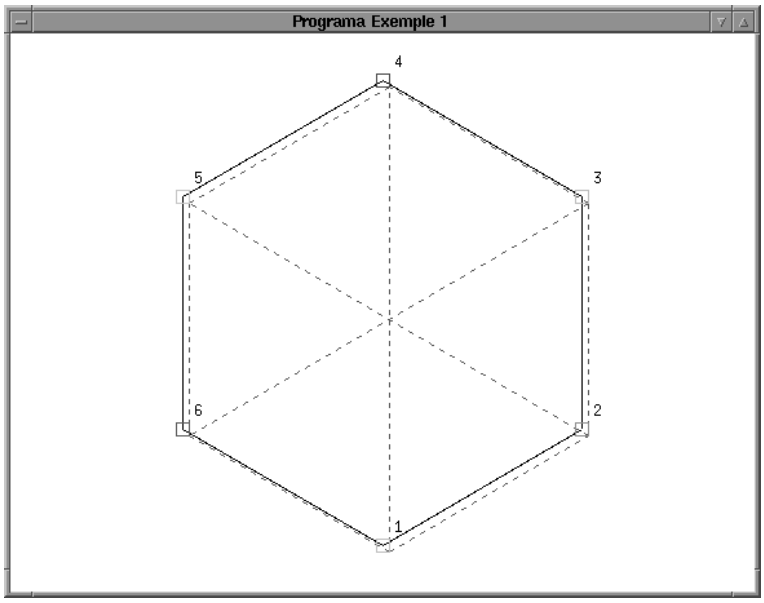

Figure 14: Initial hexagon configuration

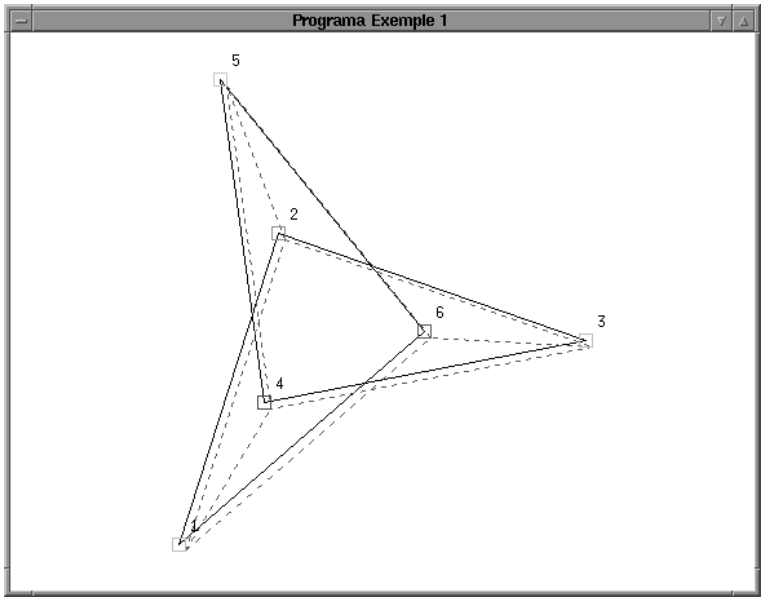

Figure 15: Final double triangular configuration

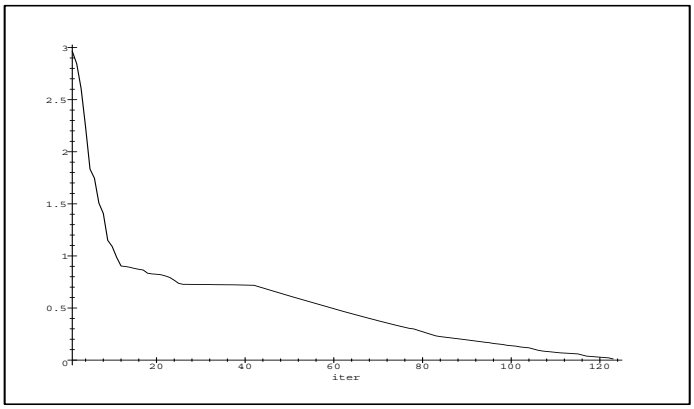

Figure 16: $||\vec{\delta}||$ evolution with global deformation

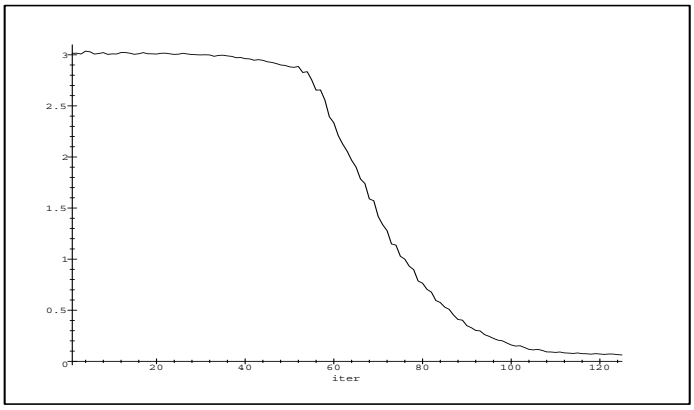

Figure 17: $||\vec{\delta}||$ evolution with one-contraint relaxation

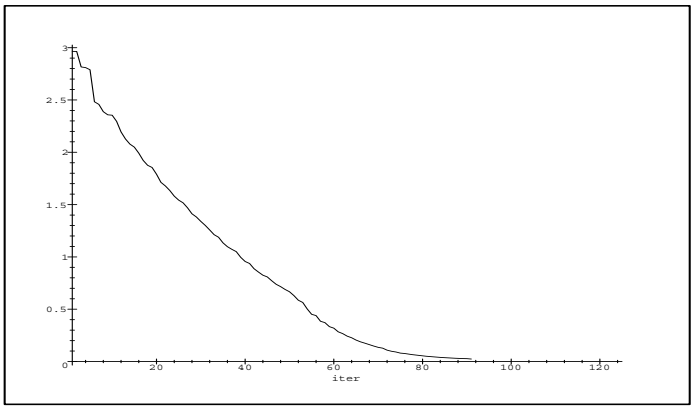

Figure 18: $||\vec{\delta}||$ evolution with hybrid method