SOLVING GEOMETRIC CONSTRAINTS BY HOMOTOPY

Hervé Lamure, Dominique Michelucci

Ecole des Mines
158 Cours Fauriel
F 42023 Saint-Etienne Cedex 02
e-mail: lamure@emse.fr, micheluc@emse.fr

Abstract

Geometric modeling by constraints yields systems of equations. They are classically solved by Newton-Raphson's iteration, from a starting guess interactively provided by the designer. However, this method may fail to converge, or may converge to an unwanted solution after a 'chaotic' behaviour. This paper claims that, in such cases, the homotopic method is much more satisfactory.

1 INTRODUCTION

In CAD, geometric modeling by constraints enables users to describe geometric objects such as points, lines, circles, conics, Bézier curves, etc in *2D* and planes, quadrics, tori, Bézier patches, etc in *3D*, by geometric constraints, *ie* distances or angles between elements, incidence or tangency relations ... This modeling yields large systems of equations, typically algebraic ones. The problem is then to solve such constraint systems.

Since the seminal work of Sutherland [Sut63], a lot of research has been done on this topic. We roughly classify resolution methods for constraint systems in three (non exclusive) categories: decomposition methods, symbolic computations, numerical algorithms.

Decomposition methods reduce constraint systems into simpler ones; solutions of irreducible subsystems are then merged. In *2D*, typical irreducible systems are triangles, the relative location of vertices of which is defined by three constraints (*e.g.* 3 distances, or 1 distance and 2 angles, ...) or systems soluble by 'ruler and compass' like Appolonius's problem; here an explicit formula does the job; however, more complex irreducible subsystems have to be solved by symbolic or numerical computations. The decomposition is performed either by the inference engine of some expert system like in Buthion [But79] or Verroust *et al* [VSR92] to cite a few, either by the matching mecanism provided by some programming language such as Prolog, as for example Brüderlin [Bru86], or by searches in graphs, like Owen [Owe91], Bouma *et al* [BFH+93], Ait *et al* [AAJM93] and many others.

Symbolic computation is used for instance by Ericson and Yap [EY88], and by Kondo [Kon92]. This method typically uses Gröbner bases, some kind of resultants, or other elimination techniques. It is exact but very slow: for instance, according to Lazard [Laz92], it is definitely impossible to compute the Gröbner bases of an irreducible polynomial system of degree 2 in 10 unknowns and variables. This kind of methods seems more suited for automatic demonstrations of geometric theorems, like for example in [CSY87, Win88].

Thus, numerical methods are essential to solve such big constraint systems met in the 'real world' of CAD. Numerical methods are used, among many others, by Borning [Bor81], Nelson [Nel85], Light *et al* [LLG81]. The most popular numerical method is Newton-Raphson's iteration and its variants. It needs an initial guess of the solution, typically given by the user thanks to some interactive tools. In an interactive application, the need for an initial guess is not really a drawback. On the contrary, it allows to select in a natural and interactive way the desired solution among an exponential number of possibilities: by Bezout's theorem, a polynomial system of total degree d, in n unknowns and equations, has $O(d^n)$ solutions. Moreover, Newton-Raphson's method is fast and works in polynomial time. Last, its use is compatible with decomposition methods (see [AAJM93] for instance) that speed up resolution.

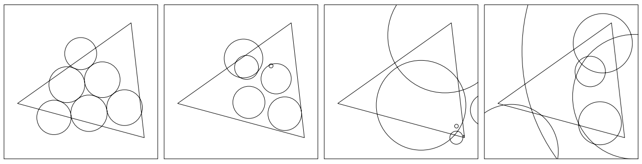

Figure 1: *Failure of Newton-Raphson's method.*

However, there is a well known problem. If Newton-Raphson's method often works fine, sometimes – much too often ! – it does not converge; or it converges to an unwanted solution after a 'chaotic' behaviour. In such a case, the user does not know what to do, apart from slightly changing his initial guess, until Newton-Raphson's method works – if it does!

This paper intends to show that the homotopic method is much more satisfactory in these situations. Its behaviour is very easy to predict, intuitive and self explanatory.

As a first argument, figure 1 shows a typical failure of Newton-Raphson's iteration, and figure 2 the behaviour of homotopy on the same example: the six circles must be tangent to each others, and must be tangent to the triangle. These images are extracted from those interactively displayed during the resolution process.

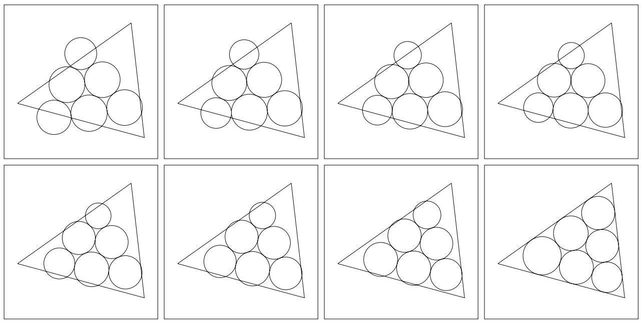

Figure 2: *Homotopy success.*

Another argument uses equation $z^3 - 1 = 0$ in $\mathbb{C}$. Of course, we do not generally need to work in $\mathbb{C}^n$ for CAD, and this equation is just a short cut for the system with two unknowns: $x, y \in \mathbb{R}^2$, and two equations: $x^3 - 3xy^2 = y^3 - 3yx^2 = 0$. Figure 3 shows attraction basins for Newton-Raphson's method and for homotopy. Basins for Newton-Raphson's method are fractals [PR86]: this explains why it is so difficult for users to predict which solution this method will converge to. On the contrary, homotopy converges to the closest[1] solution. Homotopy basins have smooth frontiers: they are semi-algebraic sets[2] when the system to be solved is algebraic: this point is detailed further in 2.5. Generally, homotopy converges much more often than Newton-Raphson's iteration to the solution intuitively closest to the initial guess, though of course this claim can not be proven in a rigorous way, since 'intuitively close' has no mathematical definition.

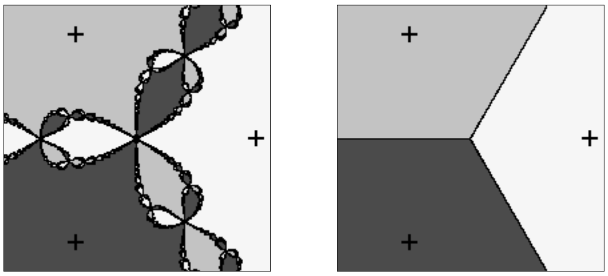

Figure 3: $z^3 - 1 = 0$, $z \in \mathbb{C}$: *Attraction basins with Newton-Raphson's method and homotopy. The +'s represent the 3 roots of unity.*

[1] In this example, attraction basins for homotopy are the cells of the Voronoï's diagram of the solution points, but this is not true in general.

[2] A semi-algebraic set is the projection of an algebraic set. An algebraic set is the solution set of a polynomial system.

This paper first gives an intuitive account on homotopy, also called continuation. Then it comments our experiments with a constraint-based geometric editor and some related questions or possible extensions. Note that we have restricted ourselves to a *2D* editor for simplicity, but homotopy, like Newton-Raphson's method, applies to all dimensions.

Previous work: resolution by homotopy is now a classical method of numerical analysis; it was already used in 1934 by Lahaye, and perhaps even before; see the recent survey by Allgower and Georg [AG93] for an history. In Computer Graphics, Dobkin *et al* [DLTW90] trace curves in $\mathbb{R}^n$ with a method inspired by a special kind of homotopy, *ie* piecewise linear approximations. In CAD, this method is used in robotics and kinematics, in particular by Morgan, Wampler and Sommese [Mor83, WMS90] who find all complex solutions of polynomial systems arising in these areas. In geometric modeling with free form surfaces, a special case of homotopy ('marching method') is also often used for following intersection curves between surfaces in $\mathbb{R}^3$ [Pat92]. Curiously enough, in modeling by constraints, it seems that until now people do not use homotopy, except a restricted continuation presented in section 2.4.4. We think that homotopy will soon supplant Newton-Raphson's method in constraint-based modelers.

2 HOMOTOPY FUNDAMENTALS

2.1 Homotopy definition

A recent survey on homotopy is [AG93].

Let $G(X) = 0$ be a system of n independant (say) polynomial equations, in n unknowns, that is $X = (x_1, x_2 ..., x_n)$ and $G = (g_1, g_2 ..., g_n)$. Well constrained systems of geometric constraints yields such systems.

Suppose now a solution $S = (s_1, s_2 ..., s_n)$ of another system $F(X) = 0$ is known, with $F = (f_1, f_2 ..., f_n)$, and that F is, in a certain meaning, 'close' to G. In our application, S is nothing else but the vector of values (vertices coordinates and circles radius) defining the initial guess interactively provided by the user, and system $F(X)$ is defined by $F(X) = G(X) - G(S)$; by construction, S is a solution of $F(X) = 0$. F and G are then embedded in a *homotopy*:

$$H(t, X) = tG(X) + (1 - t)F(X)$$

such that $H(0, X) = F(X)$ and $H(1, X) = G(X)$. System H is a linear interpolation between F and G; some homotopies use non linear interpolation, but they are beyond the scope of this paper, see [AG93].

Note: in our particular case, since $F(X) = G(X) - G(S)$, it follows that $H(t, X) = G(X) - (1 - t)G(S)$. The $(1 - t)G(S)$ term is only a function of t, and not of X. However, homotopy theory has been developed for the general case, with any F and the sequel of this presentation does not use this particularity.

System H has $n + 1$ unknowns and n equations. If $P = (t_p, X_p)$ is such that $H(P) = 0$, and if P is a regular point of $H = 0$ (*ie* the jacobian $H'(P)$ has maximal rank) then, from the implicit function theorem, $H^{-1}(0)$ is locally parametrizable by t at P. In more geometric words,

$H(t, X) = 0$ defines a curve (the *homotopy curve*) in $n+1$ dimensional space, passing through P and parametrized by t. Such a point P is known: it is $P = (0, S)$.

The main idea of resolution by homotopy is to follow the homotopy curve (also called *homotopy path*), starting from $t = 0$, $X = S$. If the homotopy path passes through a point (t, X) with $t = 1$, then a solution to $H(1, X) = 0$, and thus to $G(X) = 0$, has been found. Methods for following homotopy curves are summarized below. Figure 4 shows a sampling of 12^2 homotopy *3D* paths corresponding to our example: $z^3 - 1 = 0, z \in \mathbb{C}$ of figure 3; the checkerboard is in the plane $t = 0$.

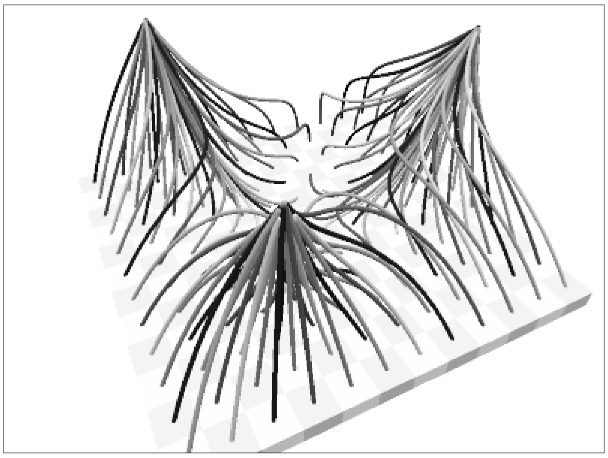

Figure 4: $z^3 - 1 = 0$, $z \in \mathbb{C}$: *Homotopy curves in 3D.*

2.2 Topologic considerations

If a homotopy curve only contains regular points, its topology is either that of a circle (*ie* the curve is a loop), or that of a line (*ie* it comes from infinity, and goes back to infinity). Of course, in each case, it may cross, or not, hyperplane with equation $t = 1$. See figure 5.

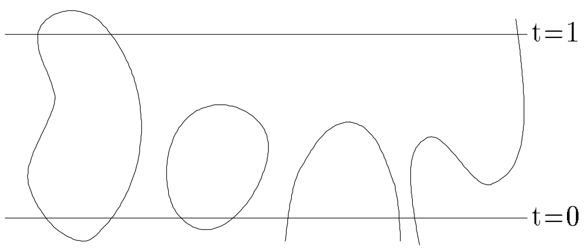

Figure 5: *Some examples of homotopy curves.*

A homotopy curve may contain *singular points*, *ie* points where the jacobian has a non-maximal rank. In the homotopy context, such points are called *bifurcation points*: two (or more) homotopy curves collide. The simplest and more frequent bifurcation points are *quadratic bifurcation points*: two curves in $\mathbb{R} \times \mathbb{C}^n$ meet in a point $Q = (t_0, X_0) \in \mathbb{R} \times \mathbb{C}^n$ where Q is a regular solution of $H(Q) = 0$ and $det(H'_X(Q)) = 0$. If ϕ is a vector tangent to one of the two curves at a quadratic bifurcation point, then $i\phi$ is the tangent vector to the second curve, where $i^2 = -1$ as usual. Moreover, the two tangent vectors have a vanishing t component. See [LW93] for a full characterization of quadratic bifurcation points.

Turning points are a special case of quadratic bifurcation points. They arise with real systems F and G: at a turning point, one of the two curves is *real*, and the other is *imaginary* or *complex*. Figure 6 shows a typical example of turning points. Turning points arise even in very simple problems, so homotopy methods must take them into account. Moreover, after [LW93], they are the only bifurcation points that arise in real sytems, in the generic case.

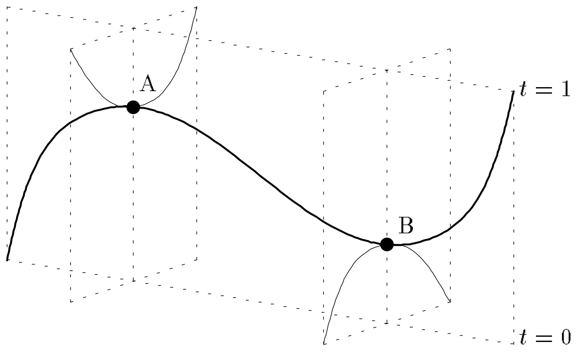

Figure 6: *A real homotopy curve, in thick line, with two turning points A and B: tangent vector has vanishing t component. In thin lines, corresponding complex branches (here, imaginary). Equations are (say) :* $F(X) = 3X^3 - X + \frac{1}{2}$, $G(X) = 3X^3 - X - \frac{1}{2}$, $H(t, X) = 3X^3 - X + \frac{1}{2} - t$, $A_t = \frac{13}{18}$, $A_X = -\frac{1}{3}$, $B_t = \frac{5}{18}$, $B_X = \frac{1}{3}$.

2.3 Homotopy method

In our case, a starting point for homotopy is known, and we are only interested by the homotopic curve crossing this point. However, in a more general setting, people want to find all (maybe complex) solutions of a given system, and have no starting points. Thus they typically proceed in two steps [AG93]. First they build a starting system of equations; it must be easily solved, and it must have at least as many solutions as the system to be solved; this number of solutions may be bounded, classically, by the product of degrees, after Bezout's theorem, or, more closely, with Newton's polytope and BKK bounds [VVC93]. Secondly, paths are followed from starting points.

2.4 Methods for following paths

2.4.1 Climbing complex homotopy

For constraint systems in CAD modeling, only real roots of system G are relevant, and the followed paths are curves in $\mathbb{R}^{n+1}$. In other areas, complex roots Z of system G are needed, and homotopy paths are curves in $\mathbb{R} \times \mathbb{C}^n$ (since, usually, t goes from 0 to 1, staying in $\mathbb{R}$). In this last case, it is possible to 'climb' along homotopy paths, starting with $(t = 0,\ Z = Z(0))$, and tracking $Z(t)$ as t monotonously increases from 0 to 1. By linearization of H at a known point (t, Z) on the curve, we get:

$$H(t+\Delta t, Z+\Delta Z) \approx H(t, Z) + H'_t \Delta t + H'_Z \Delta Z$$

We use the prediction: $\Delta Z = -(H'_Z)^{-1} H'_t \Delta t$ where Δt may possibly be scaled to control the step size $|(\Delta t, \Delta Z)|$. Then, leaving t constant (so $\Delta t = 0$), we correct several times Z by iterating: $\Delta Z := -(H'_Z)^{-1} H(t, Z)$, $Z := Z + \Delta Z$, until $|\Delta Z|$ is sufficiently close to 0.

This method faces a difficulty when a turning point, or a bifurcation point, is met. One possible solution (among others) is to use the perturbed homotopy:

$$H(t, Z) = t\gamma G(Z) + (1-t)F(Z)$$

where $\gamma = \rho e^{i\theta}$ is a random complex number: it is proven that there is only a finite number of θ for which H has bifurcation points in $t \in [0, 1)$. So, with probability 1, perturbed homotopy removes turning and bifurcation points. Of course, this perturbation can not remove possible singular solutions of $G = 0$, and corresponding bifurcation points when $H = G$, *ie* when $t = 1$.

2.4.2 Predictor-Corrector

To follow the homotopic path from a given point M_k, the so called *predictor-corrector* method first computes T_k, the tangent vector (or an approximation) to the curve in M_k, predicts that the point $P = M_k + \epsilon T_k/|T_k|$ is close to the curve, and corrects P by some variant of the Newton-Raphson's iteration (for instance using the Secant Method, or using the Moore-Penrose's pseudo inverse) or some gradient method to obtain M_{k+1}, the point on the curve closest to P. And so on, restarting from M_{k+1}. Material about numerical linear algebra can be found in [AG93]. We do not detail this method further, very similar to marching methods used in CAD for following intersection curves between surfaces in *3D* geometric modeling [Pat92, Hof90].

A practical difficulty (and a difficult theoretic problem of numerical analysis) is the choice of a good ϵ: if it is too big, the correction-step may fail; if it is too small, path following is slowed down (moreover, some numerical problems due to imprecision sometimes appear). Research has been done for safely and automatically choosing ϵ [AG93, Yak95].

However, we have found the following heuristic good enough for our limited needs and easy to implement: at each correction-prediction step, we update the pseudo inverse of H' and choose $\epsilon = 0.05$; if the correction step does not converge fast enough (*ie* in at most 4 iterations), we divide ϵ by 2 and try again. Let d the distance between the starting point M_k and the predicted point P_0; let P_1, P_2, P_3 and P_4 the successive corrections of the point P_0. As soon as distance $|P_0P_i|$ is greater than d, the system is said to diverge. It convergences when $|P_iP_{i-1}|$ is less than $0.02d$, and the angle between tangent vectors at M_k and at P_i is less than 10 degrees.

Our experimental constraint-based *2D* modeler use predictor-corrector method, mainly because we already need a Newton-Raphson's iteration to compare the behaviours of the latter and of the homotopic method. However, this method requieres the computation and (some kind of) inversion of the jacobian[3] H'. The next method only requieres to evaluate H at some points.

2.4.3 Piecewise linear approximation

Another method for following homotopy paths is known as *piecewise linear approximation*. In Computer Graphics, a variant of this method has been used for tracing curves in $\mathbb{R}^n$ by (at least) Dobkin *et al* [DLTW90].

In $\mathbb{R}^2$, this method is very simple. To trace the homotopy curve $H(t, x) = 0$, $\mathbb{R}^2$ is triangulated by, say, equilateral triangles with side ϵ. Assume a first triangle ABC traversed by H is known: H enters by the edge AB and leaves by AC because $H(A) > 0$, $H(B) < 0$, $H(C) < 0$, or the contrary. So the next triangle cut by H is ACB', where $B' = A - B + C$ is the point symmetrical to B relatively to the line AC. In ABC, H is approximated by the unique linear map $L(t, x)$ from $\mathbb{R}^2$ to $\mathbb{R}$ such that $L(A) = H(A)$, $L(B) = H(B)$ and $L(C) = H(C)$, and the curve in ABC is approximated by the edge $\{L(x, y) = 0\} \cap ABC$.

In $\mathbb{R}^{n+1}$, the space is triangulated by 'hyper tetrahedrons', *ie* simplices. Assume we know a starting simplex T traversed by the homotopy path; suppose also that values of the n functions defining homotopy H are available at the vertices of T. They define a unique linear map L from $\mathbb{R}^{n+1}$ to $\mathbb{R}^n$; in T, approximate H by L and the homotopy curve by the edge $\{L(t, x_1, ..., x_n) = 0\} \cap T$. Deduce the hyperfacet of T by which this edge leaves T, and follow it in the neighbouring simplex.

2.4.4 A restricted homotopic method

Sometimes, people in constraint-based modeling use the following restricted homotopy[Ver90]: suppose a first solution X_0 to a constraint system c_0 is known, but the value of some parameter $p \in \mathbb{R}$ (a length, an angle, a radius...) has to be changed, say from p_0 to p_1. It is convenient to see the constraint system c as a function of p, say: $c = C(p)$ and so $c_0 = C(p_0)$, and we want to solve $c_1 = C(p_1)$. When directly solving $C(p_1)$ by Newton-Raphson's iteration with X_0 as an initial guess, there is a risk of divergence. So a natural idea is to first solve, say, $C(0.9p_0 + 0.1p_1)$ with starting guess X_0 to get $X_{0.1}$ by using Newton-Raphson's method or Secant Method, then to solve $C(0.8p_0 + 0.2p_1)$ with the starting guess $X_{0.1}$ to get $X_{0.2}$, and so on, until solution X_1 of $C(p_1) = 0$ is found. Of course, this method can be generalized to any number of steps (not only 10), and for changing any number of parameters. It is a kind of naive climbing homotopy.

This method has a serious limitation: it only works when the followed path has no turning points. Since turning points are found even with very simple examples, the climbing homotopy is relevant only in the field of complex numbers (and with some kind of perturbation to avoid bifurcation points).

2.5 Attraction basins

An attraction basin for homotopy is a maximal connected points set $S \in \mathbb{R}^n$ leading to the same solution $(t = 1, X_1)$, or to no solution, when the homotopic method is applied

[3] The jacobian H' is symbolically computed for each constraint system.

with the starting point: $(t = 0, S)$. Two neighbouring basins are separated by points leading to bifurcation points. The latter are solutions of the algebraic system: $H(t, X) = det(H'_X(t, X)) = 0$ in $n + 1$ equations and unknowns, and constitute by definition an algebraic set (assuming G to be algebraic). Projection on hyperplane having equation: $t = 0$ then gives a semi-algebraic set. The reader can easily verify that, with the example: $z^3 - 1 = 0$, $z = x + iy$, frontiers between basins are (part of) lines: $y = 0$, $y = \pm x\sqrt{3}$.

It is well known that attraction basins of Newton-Raphson's resolution are fractals [PR86] (see figure 3 for instance): this explains the 'chaotic' behaviour of this method. Semi algebraic sets are less beautiful than fractals, but much smoother.

2.6 Under-constrained homotopy

For a *2D* constraint-based modeler to be truly user-friendly and interactive, we wanted to also solve under-constrained systems, and not only well-constrained ones. But, if there are n unknowns (without t) and u missing equations, homotopy $H(t, X) = 0$ no more describes a curve in $\mathbb{R}^{n+1}$, but a 'hypersurface' with dimension $1 + u$.

At the starting point $M_0 = (0, S)$, we project the 'upward' vector $V = (t = 1, x_1 = x_2 \ldots = x_n = 0)$ on the tangent space of this hypersurface to get T_0; the predicted point is then $M_0 + \epsilon T_0/|T_0|$ and a correction step gives M_1. At M_k, we project the previous tangent vector T_{k-1} on the tangent space of $H(M_k) = 0$ to get the tangent vector T_k and the predicted point is $M_k + \epsilon T_k/|T_k|$; and so on.

In other words, we try to follow the path that has an 'upward' tangent T_0 at the point $(0, S)$ and that is as straight as it can be while remaining in the homotopy surface. Up to errors (due to the fact that our ϵ is not infinitesimal), we thus follow a *geodesic curve* of the surface: in all points of such a curve, the principal normal to the curve coincide with the normal to the surface. Due to the accumulation of errors, the followed path progressively diverges from the 'true' geodesic curve; despite this limitation, the homotopy defined in this fashion has an intuitive behaviour: for instance, suppose an unknown point P must belong to a given line L, but the initial guess P_0 for P does not; then the user sees the point getting closer to the line by following the shortest path.

Another possibility we have not tried is, at each step M_k, to always project upward vector V on the tangent space of $H(M_k) = 0$; however there is a problem when $det(H'_X(M_k)) = 0$, *ie* at a local apex of H.

2.7 Solution at infinity

Solutions at infinity (for instance $xy = 1$, $y = 0$ has solution $x = \infty$) pose a problem with homotopy method, since paths to these solutions have infinite length... We have not met this problem during our experiments; anyway, it is easily (and classically) solved by homogenization. Suppose for instance unknowns are the (x, y) coordinates of some point. Just use homogenized coordinates (X, Y, H) for this point, *ie* $xH = X$, $yH = Y$, translate constraints $g_i(x, y) = 0$ into $G_i(X, Y, H) = H^{deg(g_i)} g_i(X/H, Y/H) = 0$, add a new constraint like, say: $X^2 + Y^2 + H^2 = 1$, and solve: all solutions of the homogenized system are finite; in the previous example, homogenized system is $XY = H^2$, $Y = 0$, $X^2 + Y^2 + H^2 = 1$, solution is $X = \pm 1$, $Y = H = 0$. See [AG93, Mor86].

3 A *2D* CONSTRAINT-BASED MODELER

3.1 Description

We have implemented in Lelisp an experimental *2D* constraint-based modeler to compare behaviour of resolution by Newton-Raphson's method and by homotopy. The user creates points, edges, circles in an interactive way, as with, say, MacDraw or Xfig. Moreover, he can specify constraints. Predefined constraints are: distance between two points or between a point and a line, angle between two edges, tangency between a line and a circle or between two circles, incidence between a point and a line or a circle. He can also specify (with an algebraic formula) any algebraic equation. He can declare coordinates and radius circles to be 'moveable', or not: the modeler can only modify moveable parameters. They are unknowns, and their numerical value (interactively provided by the user) are used as the initial guess by resolution methods. The user may call resolution by Newton-Raphson's method, or by homotopy. Resolution methods need constraint systems to be well- or under-constrained: so the user can incrementally add constraints and solve. Homotopy with under-constrained system is explained in section 2.6. For the moment, the modeler considers over constrained systems as mistakes from the user, and gives him a warning.

When a Newton-Raphson's resolution fails or converges to an unwanted solution, the user recovers the previous state, and uses homotopy. Generally it works much better. However, sometimes, homotopy may also converge to an unwanted solution. But homotopy is very self-explanatory: during the resolution process, intermediate states are displayed (see figure 2), so the user can easily see what is wrong in his initial guess; for instance, when he sees some circle going to the wrong side of some line, he interrupts the resolution (or waits for its end), restores the previous state, moves the circle or the line, and then calls for the resolution again: It works.

In our very first experiments, homotopy nearly always worked at the first try. Then, failures became more and more usual. The reason is that, seing homotopy work so well, we became more and more lazy and we gave initial guesses further and further away from the solution. ...

We have not met problems with bifurcation points, except when we do it on purpose, to test the software. Typically, these situations occur when there are several symmetrical solutions for a constraint system, and the initial guess has the same symmetry: for instance, if a moveable point P has two symmetrical solutions P_1 and P_2 relatively to some line L, and if the initial guess for P belongs to L, then the homotopy method can not 'choose' between the two symmetrical paths, one leading to P_1 and the other to P_2: the initial guess is itself a bifurcation point. A slight perturbation of the initial guess is sufficient to break the symmetry and to remove the bifurcation point.

3.2 Open problems

Control. When a constrained system has no solution, it is easily seen: the homotopy enters in a loop (supposing a homogenization is used to handle solution at infinity, and so to avoid infinite paths). Though feasible [Sch94], the automatic detection of loops is not implemented just now: the user has to interrupt the not ending homotopy process (his work is not lost!), or wait for the automatic stop of the process after a fixed number of steps.

Diagnosis and decomposition. Our modeler already provides some tools for diagnosis of constraint systems: recognition of under-, well-, or over-constrained unknowns by computing the Dulmage-Mendelsohn's decomposition of the bipartite graph associated to constraint systems [AAJM93], or diagnosis of rigidity like in [Hen92] for instance. These tools are essential to help users 'debug' complex constraint systems. Moreover, they allow to decompose huge constraint systems into smaller ones (by using the Konig-Hall's decomposition [AAJM93]), and so to speed up resolution. The homotopy method (like the Newton-Raphson's method, incidentally) is compatible with such decomposition methods. However, the latter are not available in our actual implemantation.

Speed. Our homotopy method is not as fast as it could be (though the constraints resolution always remains faster than their interactive specification), especially with more than 40 or 50 unknowns: when implementing, our first goal was to verify the relevance of the homotopy method, not speed... So the slowness of our implementation is not relevant. More relevant is the number of correction-prediction steps needed on average: most often, about 20 steps are enough; 60 or 70 iterations may be needed, when the followed path is very close to another homotopy curve, *ie* at 'quasi bifurcations points' (see figure 4). Thus, in practice and in average, homotopy will be 20 times slower than the Secant Method, used in each step.

Imprecision. Looking closely to attraction basins in figure 3, one can see some little errors on the frontiers: ideally, frontiers would be straigth lines (in this example). This problem is mainly due to imprecision. In some areas, this kind of problem is a severe drawback of homotopic methods, because the confusion between distinct (and very close) paths may lead to logical or topological inconsistencies; however, for our applications, and in an interactive use, this is not a serious problem.

Non algebraic equations. Homotopy is only used with algebraic constraints for the moment, though it may work with transcendental ones.

Inequalities. Inequalities are a convenient way to select a solution among others: for instance, there are two circles tangent to a given circle and passing through two given points, and an inequality can be used to select the wanted circle. Our application assumes that the choice is implicitly performed by the user, through his initial guess: our constraints only lead to equations, and not to inequalities.

However, providing explicit inequalities is perhaps useful. It is possible, at least theoretically, to translate inequalities into equations: for instance $f(X) \geq 0 \Leftrightarrow f(X) - a^2 = 0$ where a is an auxiliary real unknown: we already know how to solve an under-constrained system by homotopy. We are currently investigating this question.

4 CONCLUSION

We are convinced homotopy will soon become very popular in constraint-based geometric modelers, and maybe among descendants of MacDraw or Xfig. It is not very difficult to implement. Its behaviour is much more intuitive, predictable, and self explanatory than those of Newton-Raphson's method. It is compatible with interactivity, and with decomposition methods. As an example, we have used an experimental *2D* modeler, but homotopy (like Newton-Raphson's method) also applies for *3D* geometric constraints.

Acknowledgements

We would like to thank Jean-Michel Moreau and Marc Roelens from Ecole des Mines de Saint-Etienne for helpful discussions, and the anonymous referees for their remarks.

References

[AAJM93] S. Ait-Aoudia, R. Jegou, and D. Michelucci. Reduction of constraint systems. In *Compugraphic*, pages 83–92, Alvor, Portugal, 1993.

[AG93] E.L. Allgower and K. Georg. Continuation and path following. *Acta Numerica*, pages 1–64, 1993.

[BFH+93] W. Bouma, I. Fudos, C. Hoffmann, J. Cai, and R. Paige. A geometric constraint solver. Technical Report CSD-TR-93-054, Department of Computer Science, Purdue University, August 1993.

[Bor81] A. Borning. The programming language aspects of thinglab, a constraint oriented simulation laboratory. *ACM Transactions on Programming Languages and Systems*, 3:353–387, oct 1981.

[Bru86] B. Bruderlin. Constructing three-dimensional geometric objects defined by constraints. In *Interactive 3D Graphics*, pages 111–129, October 1986.

[But79] M. Buthion. Un programme qui résoud formellement des problèmes de constructions géométriques. *RAIRO Informatique*, 3(4):353–387, oct 1979.

[CSY87] S.C. Chou, W.F. Schelter, and J.G. Yang. Characteristic sets and grobner bases in geometry theorem proving. In *Workshop on Computer Aided Geometric Reasoning*, pages 29–56, INRIA, France, june 1987.

[DLTW90] D.P. Dobkin, S.V.F. Levy, W.P. Thurston, and A. Wilks. Contour tracing by piecewise linear approximations. *ACM Transaction on Graphics*, 9(4):389–423, oct 1990.

[EY88] L.W. Ericson and C.K. Yap. The design of linetool a geometric editor. In *Symposium on Computational Geometry*, pages 83–92, 1988.

[Hen92] B. Hendrickson. Conditions for unique realizations. *SIAM J. Computing*, 21(1):65–84, feb 1992.

[Hof90] C.M. Hoffmann. A dimensionality paradigm for surface interrogations. *Computer Aided Geometric Design*, 7:517–532, 1990.

[Kon92] K. Kondo. Algebraic method for manipulation of dimensional relationships in geometric models. *Computer Aided Design*, 24(3):141–147, mars 1992.

[Laz92] D. Lazard. Systems of algebraic equations: algorithms and complexity. Technical Report LITP 92.20, LITP, université Paris VI, VII, CNRS, mars 1992.

[LLG81] R. Light, V. Lin, and D.C. Gossard. Variational geometry in cad. *Computer Graphics*, 15(3):171–177, aug 1981.

[LW93] T.Y Li and Xiaoshen Wang. Solving real polynomial systems with real homotopies. *Mathematics of Computation*, 60(202):669–680, 1993.

[Mor83] A.P. Morgan. *Solving Polynomial Systems Using Continuation for Scientific and Engineering Problems.* Prentice-Hall, Englewood Cliffs, NJ, 1983.

[Mor86] A.P. Morgan. A transformation to avoid solutions at infinity for polynomial systems. *Applied Mathematics and Computation*, (18):77–86, 1986.

[Nel85] G. Nelson. Juno, a constraint-based graphic system. In *ACM Siggraph Conference Proceeding*, 1985.

[Owe91] J.C. Owen. Algebraic solution for geometry from dimensional constraints. In *Proceedings of the Symposium on Solid Modeling Foundations and CAD/CAM Applications*, pages 397–407, 1991.

[Pat92] N.M. Patrikalakis. Surface-to-surface intersections. *IEEE Computer Graphics and Applications*, 13(1):89–95, jan 1992.

[PR86] H.O. Peitgen and P.H. Richter. *The beauty of fractals, images of complex dynamical systems.* Springer Verlag, 1986.

[Sch94] P. Schramm. Intersection problems of parametric surfaces in cagd. *Computing*, 53:355–364, 1994.

[Sut63] I.E. Sutherland. Sketchpad, a man machine graphical communication system. In *AFIPS, Spring Joint Computer Conference*, pages 329–346, Detroit, Michigan, may 1963.

[Ver90] A. Verroust. *Etudes de problèmes liés à la définition, la visualisation et l'animation d'objets complexes en informatique graphique.* PhD thesis, Université de Paris Sud, Centre d'Orsay, 1990.

[VSR92] A. Verroust, F. Schonek, and D. Roller. Rule oriented method for parametrized computer aided design. *Computer Aided Design*, 24(3):531–540, Oct 1992.

[VVC93] J. Verschelde, P. Verlinden, and R. Cools. Homotopies exploiting newton polytopes for solving sparse polynomial systems. *SIAM J. Numerical Analysis*, 1993.

[Win88] F. Winkler. Algorithms in polynomial ideal theory and geometry. In *2nd Austrian-Hungarian Informatics Conf.*, Retzhof Austria, sep 1988.

[WMS90] C.W. Wampler, A.P. Morgan, and A.J. Sommese. Numerical continuation methods for solving polynomial systems arising in kinematics. *ASME J. on Design*, (112):59–68, 1990.

[Yak95] J.C. Yakoubsohn. An universal constant for the convergence to the newton method and application to the classical homotopy method. *Numerical Algorithms*, 1995.