Solving With Or Without Equations

Dominique Michelucci

LE2I UMR6306, CNRS, Arts et Métiers,
Bourgogne Franche-Comté University, Dijon, France

1 Equations versus algorithms, back and forth

The pentahedron problem (§2) shows the proximity between Geometric Theorem Proving (GTP) and Geometric Constraint Solving (GCS). However, the two fields separate, due to specificities of GCS (§3), which prefers algorithms to equations. Yet GCS still benefits from symbolic tools, like DAG (§4), and dual numbers (§5). Finally, §6 conjectures that algorithms can be converted to systems of equations.

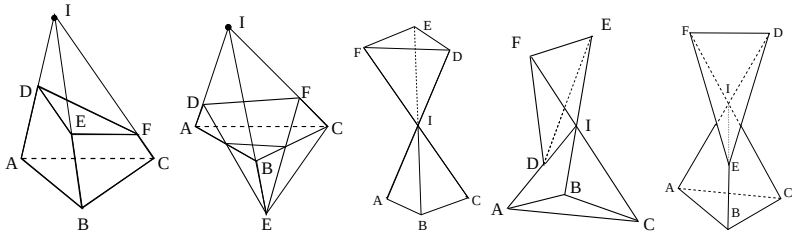

Fig. 1. Pentahedra. 6 vertices, 5 faces, 9 edges. The lengths of all edges is known.

2 The Pentahedron problem

The pentahedron problem Fig.1 [1] is to find compatible coordinates for its 6 vertices $ABCDEF$. The planarity of the 3 quadrilateral faces provides 3 constraints, and the specified lengths of the 9 edges provide 9 other constraints. This problem is well-constrained, up to 3D location and orientation. It is easy to solve for the triangle ABC and to pin it in the Oxy plane. Then it remains a polynomial system in 9 equations and 9 unknowns, the 3D coordinates of D, E, F. This system can be solved, slowly, with an interval solver. A much better formulation remarks that lines AD, BE, CF must be concurrent. Either they meet at a common point I, or they are parallel. In the first case, the 3 lengths of ID, IE, IF can be used as 3 unknowns x_{ID}, x_{IE}, x_{IF} of a smaller, and intrinsic

(coordinate-free) system of 3 equations and 3 unknowns: the law of cosines gives the 3 equations, one per quadrilateral face; *e.g.,* let $\alpha = \widehat{DIE} = \widehat{AIB}$; then

$$\cos\alpha = \frac{x_{ID}^2 + x_{IE}^2 - l_{DE}^2}{2x_{ID}x_{IE}} = \frac{(x_{ID} + l_{AD})^2 + (x_{IE} + l_{BE})^2 - l_{AB}^2}{2(x_{ID} + l_{AD})(x_{IE} + l_{BE})}$$

gives the equation in x_{ID}, x_{IE} for the quadrilateral face $ABED$. This system is solved 42 times faster than the previous one. In the second case, the point I is at infinity and a simple geometric construction shows that there are always pentahedra with parallel edges AD, BE, CF, except when some triangular or tetrahedric inequality is violated. Finally, there are 6 spurious roots, where the pentahedron is flat, so edges AD, BE, CF need not be concurrent.

This problem illustrates many common issues to GCS and GTP: what is the dimension of the manifold solution? Are there points at infinity? What is the best way to pose equations? Are there any degenerate solutions, and what is the topological dimension of the degenerate manifold? Indeed, in GTP, non degeneracy conditions (the triangle must not be flat, vertices must be distinct, etc) have to be specified in order to prove theorems. This example also shows that GCS and GTP are close while all constraints are incidence, or distance, or angle constraints between flats: points, lines, planes. But the latter are not sufficient in CADCAM.

3 Specificities of GCS for CADCAM

The first specificity in GCS is the inaccuracy issue. The nullity of a number, the equality of two numbers are no more decidable. The computation of the rank of a set of vectors, or of a Jacobian, is no more guaranteed. The distinction between $x > 0$ and $x \geq 0$ becomes irrelevant. The equivalence $x \neq 0 \Leftrightarrow \exists y \,|\, xy - 1 = 0$ used in Gröbner bases becomes irrelevant as well.

Another specificity is the need for optimization and algorithms.

For example, there are many orthogonal projections of a point on a non linear curve or surface but for distance constraints, only one is relevant. First order conditions, like KKT (Karush-Kuhn-Tucker), are necessary but not sufficient to fully characterize solutions. Solving KKT equations provides a superset of the roots, and spurious ones (saddle points, local optima) must be cancelled with some algorithm.

When computing the orthogonal projection of a point p to a composite object (*e.g.,* the union of a line and a conic), often used in CADCAM, the relevant system of equations depends on the location of p. Again, some optimization problem occurs. For the orthogonal projection on a part of an object, like a segment, optimization can be avoided, but an algorithm and some if-then-else are more convenient than equations.

CADCAM systematically uses piecewise polynomials (box splines, bsplines, etc). They are not polynomials and standard tools of Computer Algebra (Gröbner bases, Wu-Ritt method, resultants, GCD, fundamental theorem of algebra, Sturm's theorem, etc) no more apply. Idem for NURBS and piecewise rational functions.

Finally, Computer Graphics and CADCAM use algorithmic shapes, called features or parametric objects, like staircases, gears and sprockets, etc. The number of steps in a staircase is an integer (thus diophantine equations occur) and depend on parameter values of length and height: the number of unknowns and equations depend on parameter values. In passing, there is some similarity with Steiner's porism, or Poncelet's porism, in GTP.

Worse, subdivision curves and subdivision surface have invaded Computer Graphics: designers interactively define a coarse mesh, and a procedure rounds vertices and edges. Most of the time, there is no equation for the limit surface.

Sometimes, equations are available but too huge to be symbolically expanded, *e.g.,* $\det(M(X)) = 0$. Of course, a numerical algorithm can still compute $\det(M(V))$ for a given numerical vector V. Another example is given by intersection curves between rational surfaces: they are not rational but all geometric modelers approximate them with rational curves.

4 DAGs

Gouaty et al [2] solve such geometric constraints for CADCAM: equations are replaced with algorithms. Constraints are represented with DAGs (Directed Acyclic Graph). DAG is a popular data structure in Dynamic Geometry softwares (where they are called Straight Line Programs) and in Computer Algebra. In CADCAM, DAGs involve spline or NURBS functions, algorithms (for rounding, for orthogonal projection), subdivision surfaces and other algorithmic shapes. They are no more convertible into polynomials, and it is no more possible to compute the DAG of the derivative of a given DAG. But these DAG keep some interesting features: they still can be evaluated for given values of parameters, thus it is still possible to solve; DAG can be interactively specified and modified by users or designers who are not computer scientists, thus users can still pose their problems; probabilistic tests for nullity or equality (up to some tolerance) are still possible; and finally, exact computations (up to floating point precision) of derivatives are still possible, after all, with dual numbers. This is interesting because derivatives computed with finite differences are inaccurate, which hampers the convergence of numeric solvers close to the solution.

5 Dual numbers

The idea is to attach an infinitesimal number ϵ_i to each unknown x_i, with the rule $\epsilon_i^2 = \epsilon_i \epsilon_j = 0$. The addition is straightforward. The product, for one ϵ, is given by:

$$\begin{array}{ccccc} (a+b\,\epsilon) & \times & (a'+b'\,\epsilon) & = & aa' + (ab'+ba')\,\epsilon \\ \downarrow & & \downarrow & & \downarrow \\ \begin{pmatrix} a & 0 \\ b & a \end{pmatrix} & \times & \begin{pmatrix} a' & 0 \\ b' & a' \end{pmatrix} & = & \begin{pmatrix} aa' & 0 \\ ba'+ab' & aa' \end{pmatrix} \end{array} \tag{1}$$

and it is generalizable to many ϵ_i. The bijection between dual numbers and matrices is an isomorphism: the matrice of the opposite (inverse) of a dual number is the opposite (inverse) of the matrice of the dual number. Other rules are:

$$\frac{1}{a+b\,\epsilon} = \frac{1}{a} - \frac{b}{a^2}\,\epsilon \text{ when } a \neq 0 \tag{2}$$

thus $b\epsilon$ has no inverse (the associated matrice is not invertible). This rule is a special case of:

$$(a+b\epsilon)^k = a^k + ka^{k-1}b\,\epsilon \tag{3}$$

If P is a polynomial, then $P(x_v + \epsilon)$ where x_v is a floating-point number, gives $P(x_v)$ and the derivative $P'(x_v)$:

$$\begin{aligned} P(x_v + \epsilon) &= a(x_v+\epsilon)^3 + b(x_v+\epsilon)^2 + c(x_v+\epsilon) + d \\ &= a(x_v^3 + 3x_v^2\,\epsilon) + b(x_v^2 + 2x_v\,\epsilon) + c(x_v + \epsilon) + d \\ &= (ax_v^3 + bx_v^2 + cx_v + d) + (3ax_v^2 + 2bx_v + c)\,\epsilon \\ &= P(x_v) + P'(x_v)\,\epsilon \end{aligned} \tag{4}$$

It extends to multivariate polynomials: either we have only one ϵ and two evaluations are needed:

$$\begin{aligned} Q(x_v+\epsilon, y_v) &= Q(x_v, y_v) + Q'_x(x_v, y_v)\epsilon \\ Q(x_v, y_v+\epsilon) &= Q(x_v, y_v) + Q'_y(x_v, y_v)\epsilon \end{aligned} \tag{5}$$

or each variable is attached its own ϵ and one evaluation suffices:

$$Q(x_v+\epsilon_x, y_v+\epsilon_y) = Q(x_v, y_v) + Q'_x(x_v, y_v)\epsilon_x + Q'_y(x_v, y_v)\epsilon_y$$

Dual numbers extend to non polynomial functions:

$$\begin{aligned} \exp(a+b\,\epsilon) &= e^a + be^a\,\epsilon \\ \cos(a+b\epsilon) &= \cos(a) - b\sin(a)\,\epsilon \\ \sin(a+b\epsilon) &= \sin(a) + b\cos(a)\,\epsilon \\ \tan(a+b\epsilon) &= \tan(a) + b(1+\tan^2(a))\,\epsilon \\ |a+b\epsilon| &= |a| + (\mathrm{sgn}(a)b + (1-\mathrm{sgn}(a)^2)|b|)\,\epsilon \end{aligned}$$

Dual numbers permit to compute the derivative of $D(X) = \det(M(X))$, for square matrices $M(X)$, even if entries of M are piecewise polynomials, or algorithms: just replace floating point numbers with dual numbers and then use any standard numerical method (Gauss pivot, LUP). There are also formulas.

Lemma: $\det(I+\epsilon M) = 1 + \mathrm{Trace}(M)\,\epsilon$, where $M \in \mathbb{R}^{n,n}$. Proof:

$$\det(I+\epsilon M) = (1+M_{11}\epsilon)(1+M_{22}\epsilon)\dots(1+M_{nn}\epsilon) + R = 1 + \mathrm{Trace}(M)\,\epsilon + R$$

where R represents other perfect matchings in $I+\epsilon M$. But other matchings use at least two off-diagonal entries in $I+\epsilon M$, thus are multiples of ϵ^2, thus are zero.

When A is inversible, $\det(M(x+\epsilon)) = \det(A+\epsilon B)$ is:

$$\begin{aligned} \det(A+\epsilon B) &= \det(A(I+\epsilon A^{-1}B)) \\ &= \det(A)\det(I+\epsilon A^{-1}B) \\ &= \det(A)(1+\mathrm{Trace}(A^{-1}B)\,\epsilon) \end{aligned} \tag{6}$$

When A is not inversible, we use its SVD : $A = U\Sigma V^t$ (with Σ diagonal and U, V unitary):

$$\begin{aligned} \det(A+\epsilon B) &= \det(U\Sigma V^t + \epsilon B) \\ &= \det(U(\Sigma V^t + \epsilon U^t B)) \\ &= \det(U(\Sigma + \epsilon U^t B V)V^t) \\ &= \det(\Sigma + \epsilon U^t B V) \end{aligned} \tag{7}$$

equals the product of diagonal entries of $\Sigma + \epsilon U^t B V$. It is 0 when there are at least two null singular values in Σ. Otherwise it is

$$(\sigma_1 + k_1\epsilon)\ldots(\sigma_{n-1}+k_{n-1}\epsilon)(0+k_n\epsilon) = 0 + \sigma_1\ldots\sigma_{n-1}k_n\,\epsilon \tag{8}$$

The extension to many ϵ is lengthy but easy.

Dual numbers provide exact (up to floating point precision) derivatives even when equations are not available and are replaced with algorithms. Thus they make possible to use Newton method for solving, Euler method for following an homotopy curve, BFGS method for optimizing.

It is possible to compute Taylor expansions beyond degree 1 (using $\epsilon^4 = 0$), which eases Runge Kutta method for homotopy. It has a cost, reducible with the sparsity of ϵ expansions. ϵ expansions are sortable [3] with compatible orders used in Gröbner bases.

In passing, an algebraic construction ϕ starting from $\mathbb{R}$ gives the quaternions, which represent 3D rotations. If ϕ is applied to $\mathbb{R}+\epsilon\mathbb{R}$, it gives biquaternions, aka dual quaternions, which represent both 3D rotations and translations.

6 From algorithms to systems of equations

Algorithms are more convenient than equations to express constraints. But maybe algorithms can be automatically converted into systems of equations, and algorithms are just a convenience to pose equations?

Let $a \in \mathbb{R}$, and $s = \mathrm{sgn}(a)$ be the sign of a: $a = 0 \Rightarrow s = 0$, $a \neq 0 \Rightarrow s = |a|/a$. Then the system of equations below is such that $S(a,s) = 0 \Leftrightarrow s = \mathrm{sgn}(a)$.

$$\begin{cases} 0 = s^3 - s \quad \Leftrightarrow s \in \{0,-1,1\} \\ 0 = a - sy^2 \Leftrightarrow y^2 = |a| \text{ except when } a = 0 \\ 0 = y^2 z - 1 \Leftrightarrow y^2 \neq 0 \text{ Remark that } 0 = yz - 1 \text{ also works} \end{cases}$$

The reader can check that when $a > 0$, there is only one real solution $y^2 = |a| = a$, $s = 1$, $z = 1/a$. When $a < 0$, there is only one real solution $y^2 = |a| = -a$, $s = -1$, $z = -1/a$. Finally, if $a = 0$, then $s = 0$ and y^2 is free; for uniqueness, add the equation: $(1-s^2)(y-1) = 0$. It changes nothing if $a \neq 0$.

Otherwise, if $a = 0$, the only real solution is $s = 0, y = z = 1$. Then it is easy to build systems $S(a, R)$ for defining $|a|$, the positive (negative) part a^+ (a^-), etc:

$$
\begin{array}{l}
R = |a| = sa \\
R = a^+ = \max(0, a) = (a + |a|)/2 = (a + sa)/2 \\
R = a^- = \min(0, a) = a - \max(0, a) = (a - |a|)/2 = (a - sa)/2 \\
R = \max(a, b) = (a + b)/2 + |b - a|/2 \\
R = \min(a, b) = (a + b)/2 - |b - a|/2
\end{array}
$$

Then we can convert the if-then-else instruction: $F =$ (if $x > 0$ then P else if $x < 0$ then N else Z) into equations:

$$F = \mathrm{sgn}(x^+)P + \mathrm{sgn}(x^-)N + (1 - (\mathrm{sgn}(x^+) + \mathrm{sgn}(x^-))Z$$

For translating the arithmetic constraint $x \in \mathbb{Z}$ into equations, we can use the equation $\sin(\pi x) = 0$, which indeed describes $\mathbb{Z}$, but it is not algebraic. An algebraic system is: $x = x_0 + 2x_1 + \ldots 2^n x_n$ and $x_i(1 - x_i) = 0$ for all $i \in [[0, n]]$. The system has logarithmic size in 2^n, but it describes only integers in $[[0, 2^n - 1]]$. The naive representation: $x(x - 1) \ldots (x - 2^n - 1) = 0$ is exponential size.

After functional programming, assignments and iterations are useless. It is sufficient to consider fixpoints : $F(F(...(X)))$, where F is some algorithm. Assume the program $Y = F(X)$ is represented with some system of equations: $S(X, Y) = 0$. Then fixpoints of F are solutions of the system $S(X, X) = 0$. The latter clearly shows that the sizes of X and Y must be equal. It is untrue for some algorithm F, *e.g.,* for subdivision curves, the size of Y is twice the size of X.

We only sketched this conversion: we did not treat the conversion of function calls, or of data structures like Lisp pairs. Assume this conversion is possible under mild assumption. Then Computer Algebra applies to resulting polynomial systems, *e.g.,* ideals and radicals concepts become relevant for piecewise polynomials after all. We can deduce equations from algorithms, *e.g.,* for the distance between a point and a segment, or for canceling spurious roots. Thus algorithms are just a convenient way to pose equations. Is it possible to find (Gröbner bases of) polynomial preconditions for the algorithm to work, and to fail?

References

1. Barki, H., Cane, J.M., Garnier, L., Michelucci, D., Foufou, S.: Solving the pentahedron problem. Computer-Aided Design 58, 200–209 (2015)
2. Gouaty, G., Fang, L., Michelucci, D., Daniel, M., Pernot, J.P., Raffin, R., Lanquetin, S., Neveu, M.: Variational geometric modeling with black box constraints and dags. Computer-Aided Design 75, 1–12 (2016)
3. Michelucci, D.: An epsilon-arithmetic for removing degeneracies. In: 12th Symposium on Computer Arithmetic. p. 230. IEEE (1995)