Solving with or without equations

D. Michelucci

ADG 2016, Strasbourg

June 25, 2016

Plan of the talk

The **pentahedron** problem shows the proximity btw Geometric Theorem Proving and Geometric Constraint Solving

The two fields separate: **specificities of GCS**, which goes **from equations to algorithms**.

GCS examples in CADCAM.

GCS still benefits from symbolic tools, like **DAG**, and **dual numbers**.

From algorithms to equations: what if algorithms were just user-friendly way to pose equations, after all?

Pentahedron problem

Many common issues in both GTP & GCS:

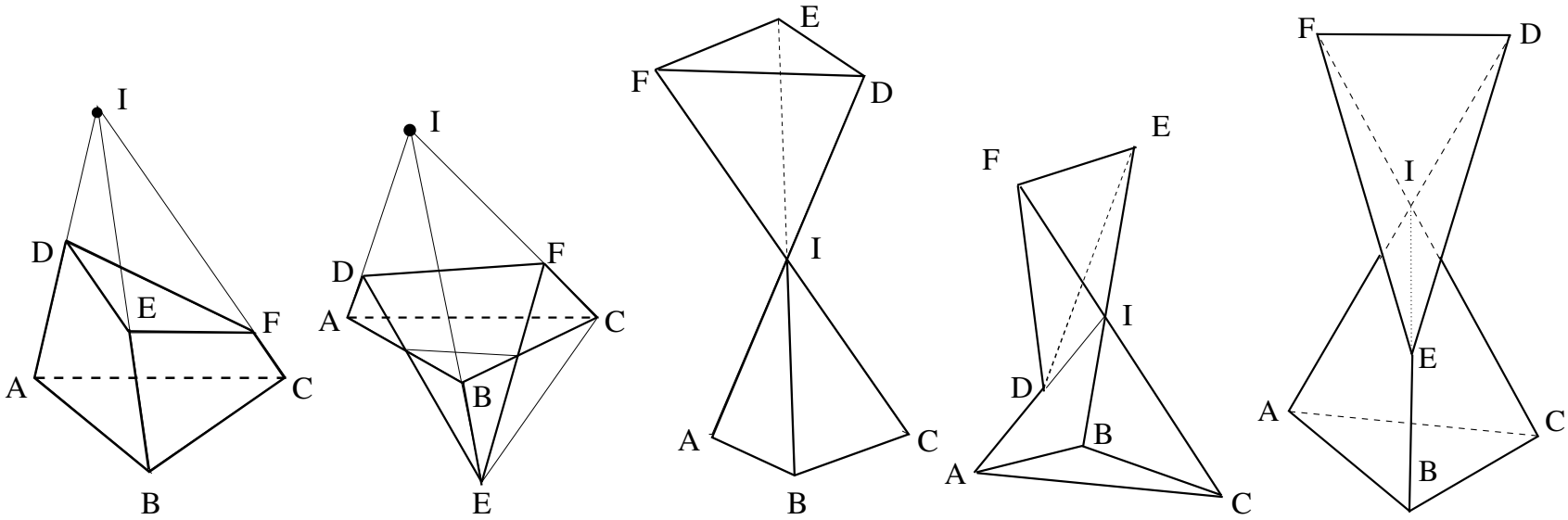
- dimension of the solution manifold,

- manifolds of spurious (degenerate) roots,

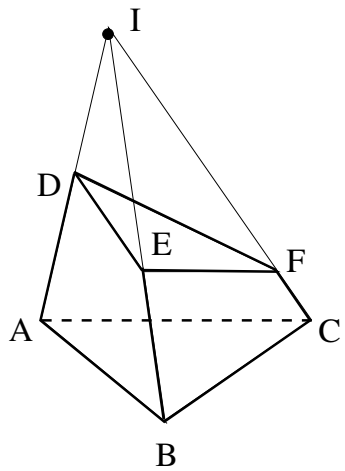
- points at infinity,

- many ways to pose equations.

Pentahedron problem

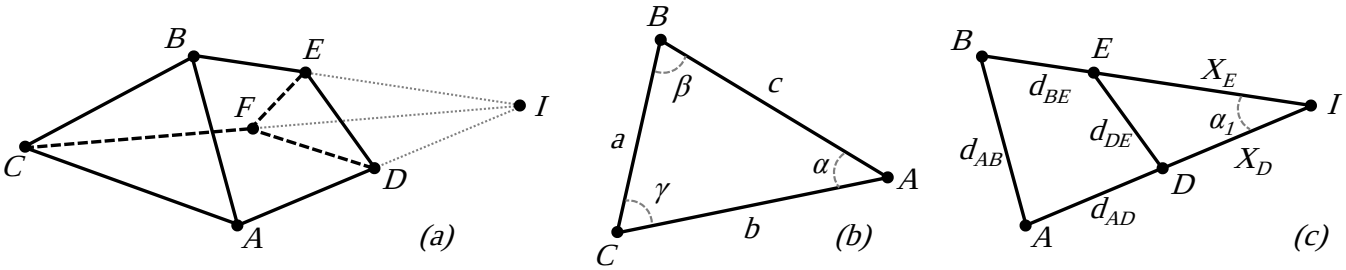


Pentahedron problem



First formulation: fix ABC in Oxy $\Rightarrow$ 9 unknowns and equations: coplanarities of 3 quadrilateral faces and 6 pt-pt distances.

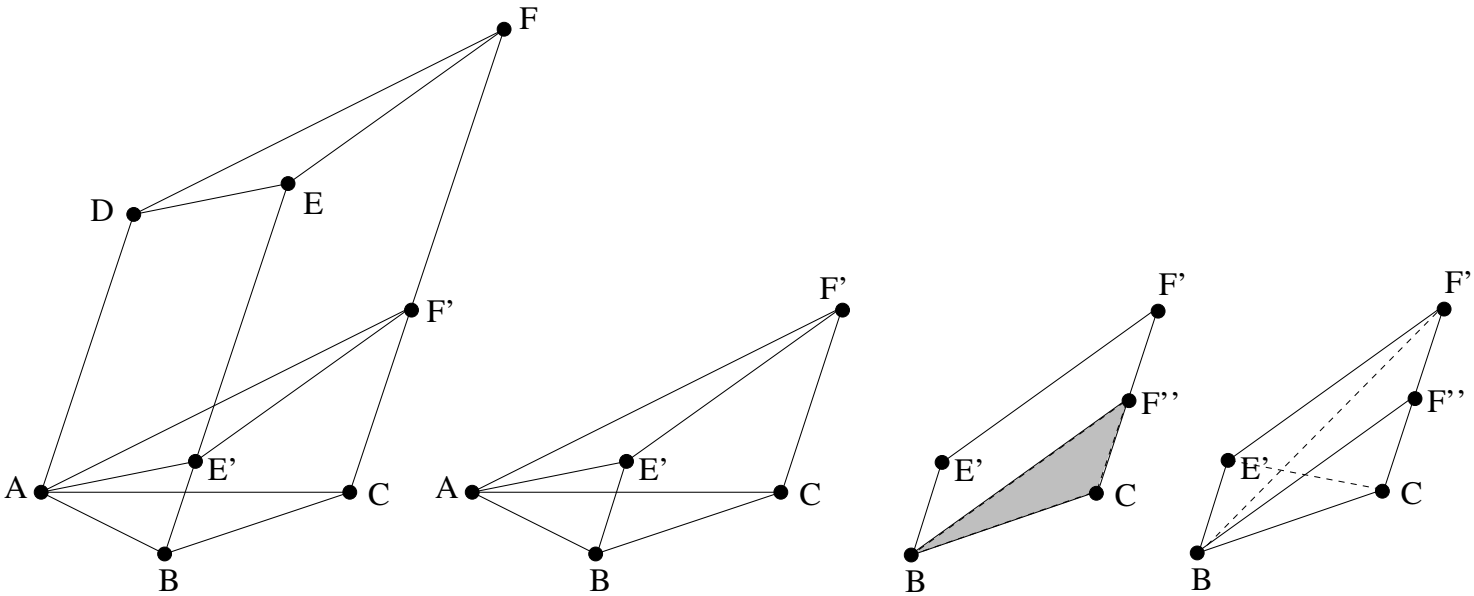
Pentahedron problem: I is not at infinity



$$\cos\alpha = \frac{x_D^2 + x_E^2 - d_{DE}^2}{2x_D x_E} = \frac{(x_D + d_{AD})^2 + (x_E + d_{BE})^2 - d_{AB}^2}{2(x_D + d_{AD})(x_E + d_{BE})}$$

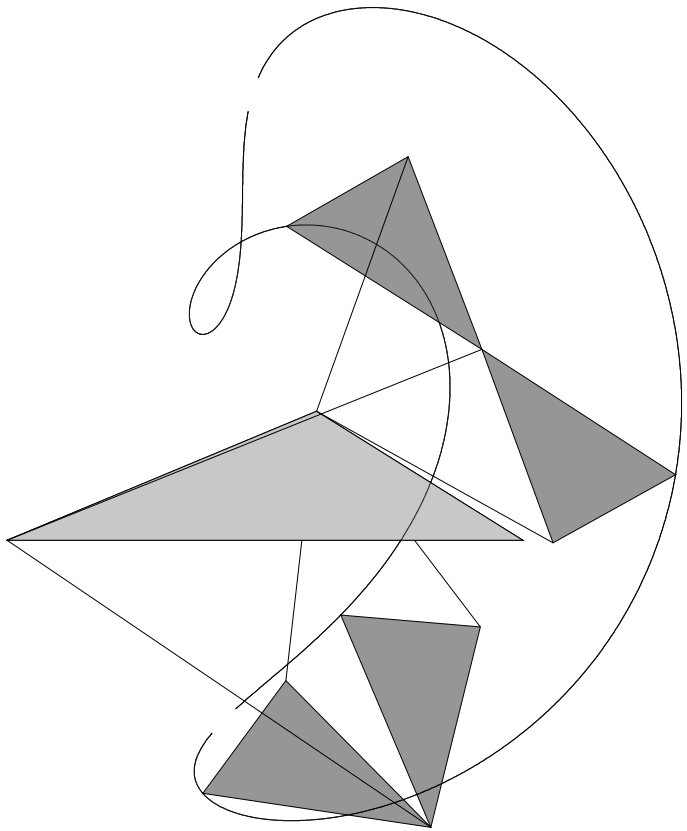
Better **coordinates-free** formulation, 3 times smaller. 40 times faster to solve with intervals. 3 unknowns are lengths ID, IE, IF. 3 relations for angle at I.

Pentahedron pbm: parallel solutions



There is almost always parallel solutions! and an easy geometric construction.

Pentahedron pbm: spurious roots, flat pentahedra



There is a finite number of spurious roots: **flat pentahedra**. Pin ABC, forget constraint CF: DEF can move around ABC. At most 6 roots (intersection between sextic curve and circle).

Pentahedron problem, manifold of spurious roots

Hexaedron or dodecahedron: a manifold of flat solutions.

Known difficulty in GTP: specify non degeneracy conditions in order to prove geometric theorems

Numerical analysis: deflation methods

Interval Analysis: the problem seems less known. Hint: search the root closest to a given point: the solution set is discrete.

The pentahedron problem

The pentahedron problem shows that:

GCS and GTP are very close while all constraints are:

incidence / distance / angle constraints between points / lines /planes.

But it is not sufficient for CADCAM, and GCS have **specificities**.

Plan of the talk

The **pentahedron** problem shows the proximity btw Geometric Theorem Proving and Geometric Constraint Solving

The two fields separate: **specificities of GCS**, which goes **from equations to algorithms**.

GCS examples in CADCAM.

GCS still benefits from symbolic tools, like **DAG**, and **dual numbers**.

From algorithms to equations: what if algorithms were just user-friendly way to pose equations, after all?

Specificities: 1, Inaccuracy issue

Even with the simplest constraints, there are issues / troubles specific to GCS:

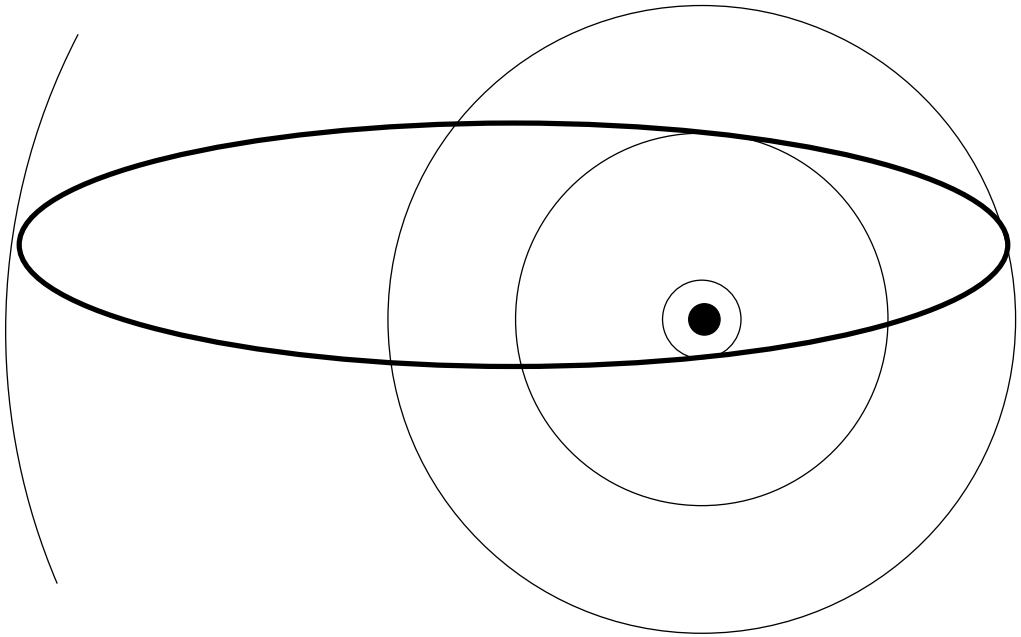
Inaccuracy issue

$\Rightarrow$ hard to compute the rank of Jacobians

$\Rightarrow$ distinction between $x > 0$ and $x \geq 0$ is irrelevant

$\Rightarrow$ the trick $x \neq 0 \Leftrightarrow xy - 1 = 0$ (where y is auxiliary) used in Grobner Bases makes no sense

Specificities: 2, need for optimization

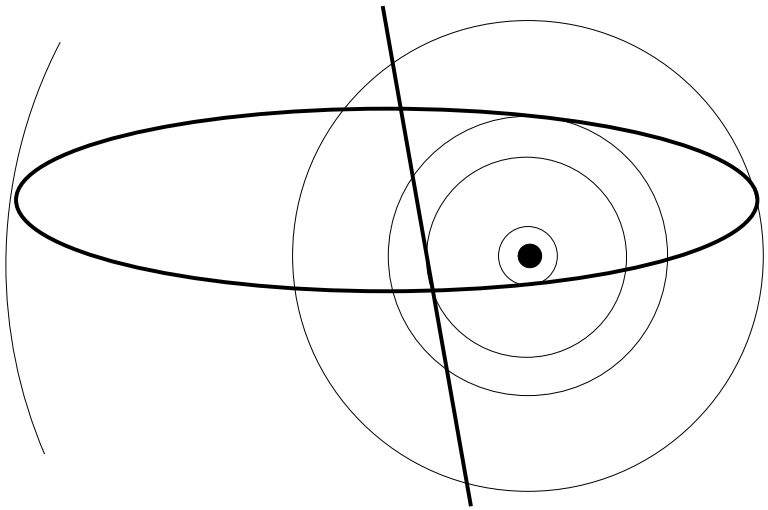


Many orthogonal projections of a point on a curve/surface. Thus an optimization problem arises.

KKT (or FJ) are necessary but not sufficient to fully characterize solutions. An **algorithm** is needed to cancel spurious roots.

Specificities: 3, composite objects

Several systems for distance to a composite object. A minimization problem. **Equations depend on the location.**

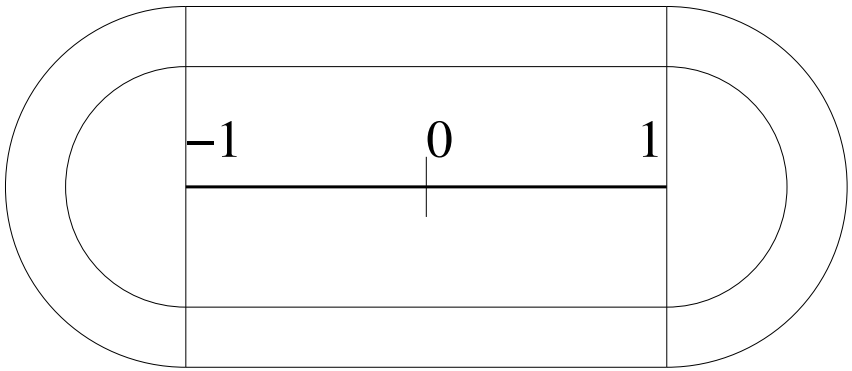


Spatial dictionary, or geo-referenced systems of equations: a 3D bounding box is recursively subdivided, and boxes at the leaves of the tree contain the active system of equations, and other geometric data (*e.g.*, a simplified CSG, a mesh). Ex: ray-casting.

Specificities: 4, algorithms are very convenient

It is much easier to compute the distance with an algorithm and if-then-else.

Which equation for distance to a segment?



Specificities: 5, piecewise polynomials are not polynomials

In CADCAM, *splines provide very convenient bases for PP, **piecewise** polynomials. But PP are not polynomials:

$\Rightarrow$ you can not use resultants, discriminants, GCD, ideals and radicals, Groebner bases, Wu-Ritt, Gauss theorems (fundamental thm of algebra, + gcd-primitive-content-parts), Sturm theorem (counting real roots), the proof that some homotopy finds all roots, etc

Idem for NURBS, bases for **piecewise rational** curves/surfaces.

Specificities: *splines versus polynomials

+ geometric bases provide tight bounding boxes (convex hull property) and subdivision: very convenient for solving *e.g.*, with Bernstein polynomials:

$$P(x \in [0,1], y \in [0,1]) = \sum_i \sum_j c_{i,j} B_i(x) B_j(y) \quad \in \ [\min c_{ij}, \max c_{ij}]$$

- unfortunately, equations which are sparse in the canonical base are dense in geometrical bases (*e.g.*, tensorial Bernstein bases). There is an exponential number of basis functions and finding the smallest/greatest coefficient is NP-hard (Kubicki et al).

Specificities: The simplicial Bernstein basis

The number of basis functions in the simplicial (or barycentric) Bernstein basis is polynomial, but few papers investigate that.

Reuter, Mikkelsen, Sherbrooke, Maekawa & Patrikalakis: Solving nonlinear polynomial systems in the barycentric Bernstein basis. Visual Computer 24.3 (2007) : 187-200.

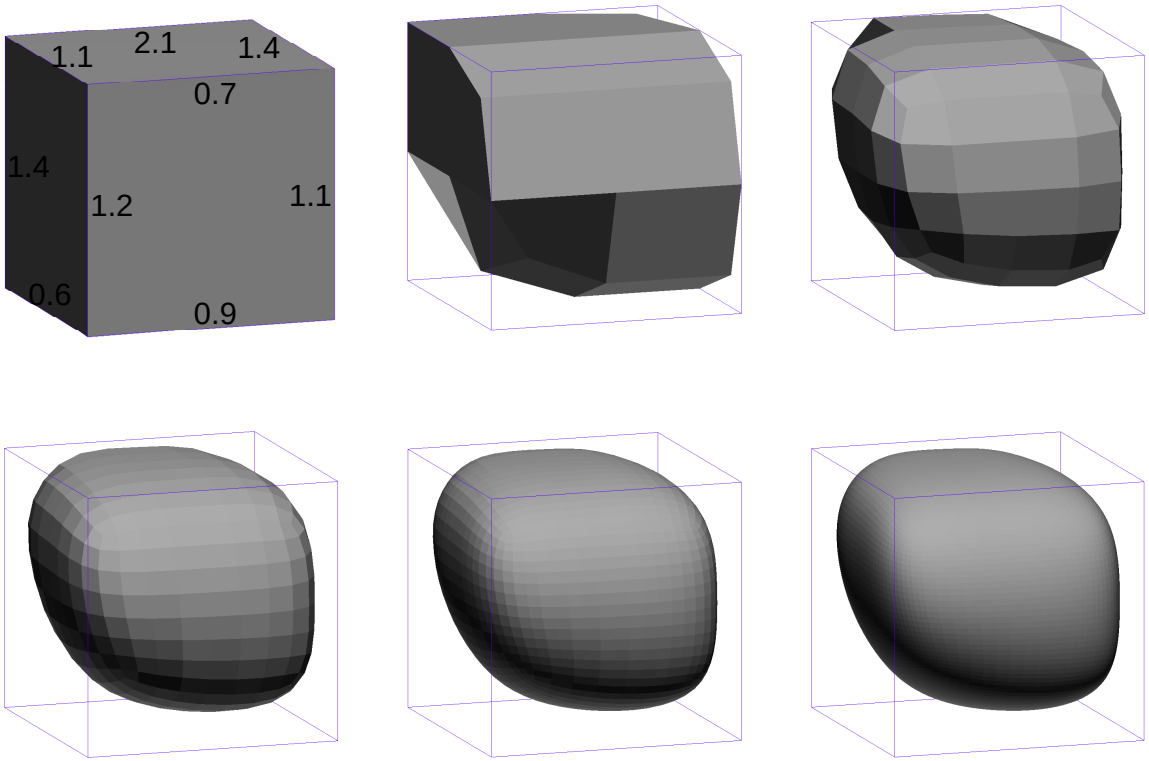
Specificities: 6, algorithmic objects/shapes

Algorithms use assignments, if-then-else, iterations, recursions, fix-points, other (ODE, PDE, etc) solvers.

They are used to define algorithmic objects ("parametric objects", "features"):

ruler and compass constructions, ok, but also:

Specificities: algorithmic objects like subdivision surfaces



5 iterations of an irregular subdivision surface with weighted edges

Specificities: algorithmic objects/shapes

subdivision curves/surfaces: **often no equation is available**.

- staircase, ladder,

- gears and sprockets

- cars, buildings

- fractals (fractal antenna)

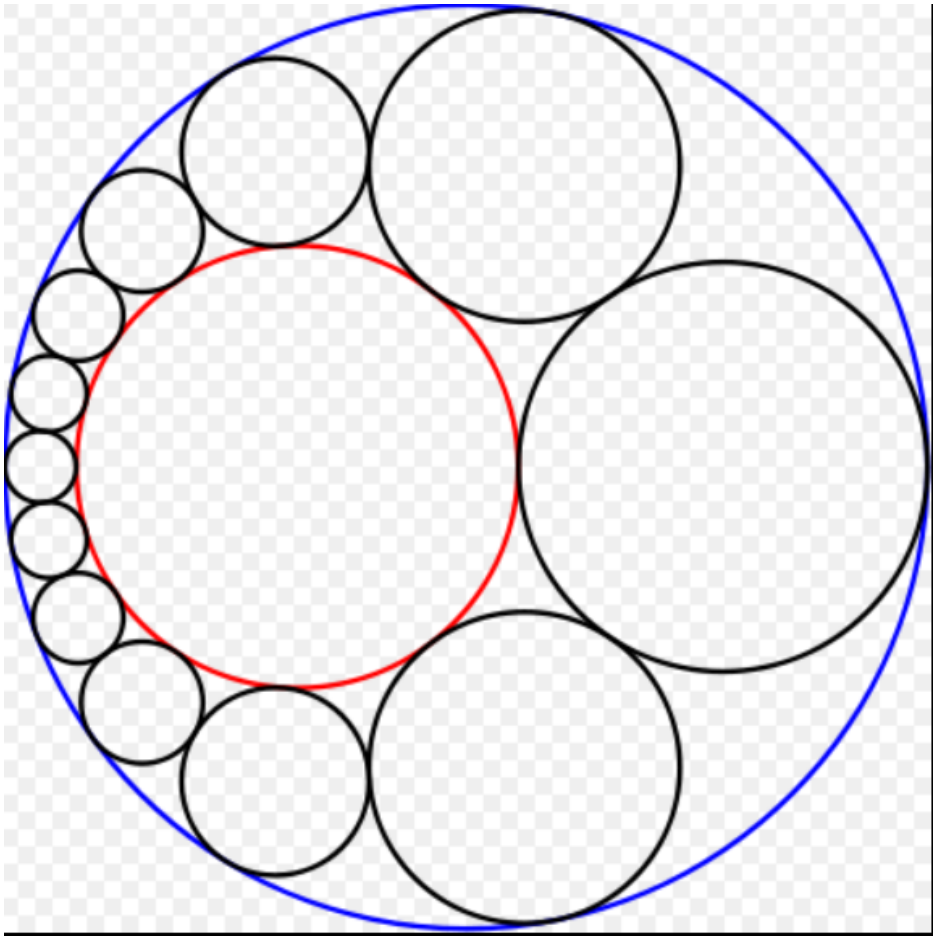
Specificities: features of algorithmic shapes

The number of steps, and of intermediate stages, in a staircase, depend on parameter values (length, height), and is **integer**.

$\Rightarrow$ **the system depends on parameter values** (*e.g.*, number of unknowns & equations)

Remember Matiyasevich' answer to Hilbert's 10th pbm: no algorithm to solve general diophantine equations.

(It reminds Steiner's porism and Poncelet's)



Sketch of the proof of Steiner's or Poncelet's thm

Obvious when the inner and outer circles are concentric.

Inversion preserves lines and cercles.

There is always an inversion to make two cercles concentric.

Btw, how to prove that for any number of inbetween circles, with Groebner bases or ascending/regular chains?

Specificities: 7, case without practical equation

$$|M(X)| = 0$$

For big or not so big but dense M, the determinant can not be expanded. But, it is easy to evaluate $|M(V)|$.

There is a small equivalent system: introduce unknowns $\lambda_1, \ldots \lambda_n$ and equations $1 = \sum \lambda_i^2$ and $\forall c$, $0 = \sum_l \lambda_l M_{l,c}$.

It is possible to compute derivatives $\partial |M(X)| / \partial X_k$ for $X = V$ with dual numbers (below).

Actually, $|M(X)| \neq 0$ is harder, for numerical solver.

Specificities: when algorithms replace equations

Distance / incidence between a given point and a given surface

- subdivision surface: no equation available; spurious roots

- implicit or parametric surface: equations are known but maybe PP; spurious solutions

Remark: the surface can be unknown, and the point as well.

Specificities: geometric algorithms

For known curves/surfaces, geometric algorithms are used. They rely on subdivision (de Casteljau, Oslo, etc) and Branch and Bound methods to prune the search space. They remove spurious solutions.

Thus, no equation for: $\text{dist}(p, S) = 0$ or $x_5 - \text{dist}(p, S) = 0$ but algorithms to compute $\text{dist}(p, S)$ for given p, S.

Specificities: when (PP) equations available but irrelevant

CAD: Intersection curves between rational surfaces are not rational curves.

But they are approximated with rational curves, for methods to be re-entrant, in all CADCAM geometric modelers.

Plan of the talk

The **pentahedron** problem shows the proximity btw Geometric Theorem Proving and Geometric Constraint Solving

The two fields separate: **specificities of GCS**, which goes **from equations to algorithms**.

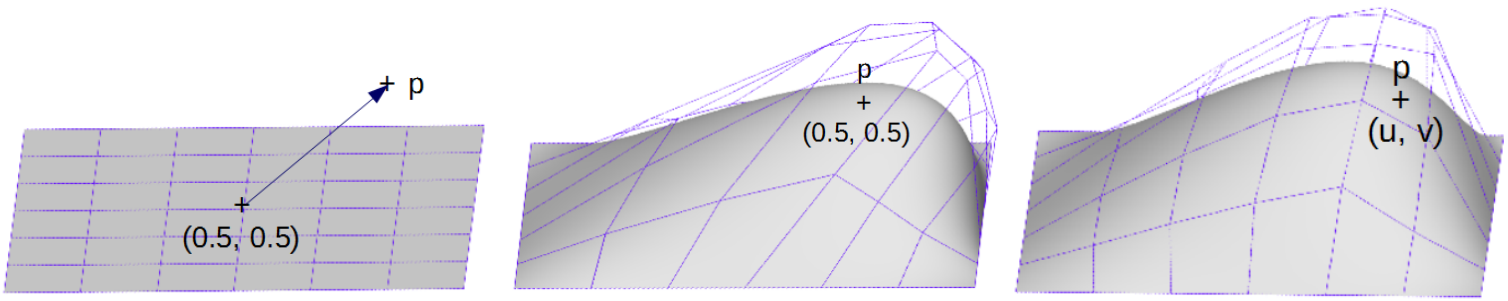
GCS examples in CADCAM.

GCS still benefits from symbolic tools, like **DAG**, and **dual numbers**.

From algorithms to equations: what if algorithms were just user-friendly way to pose equations, after all?

GCS Example 1/3 in CADCAM

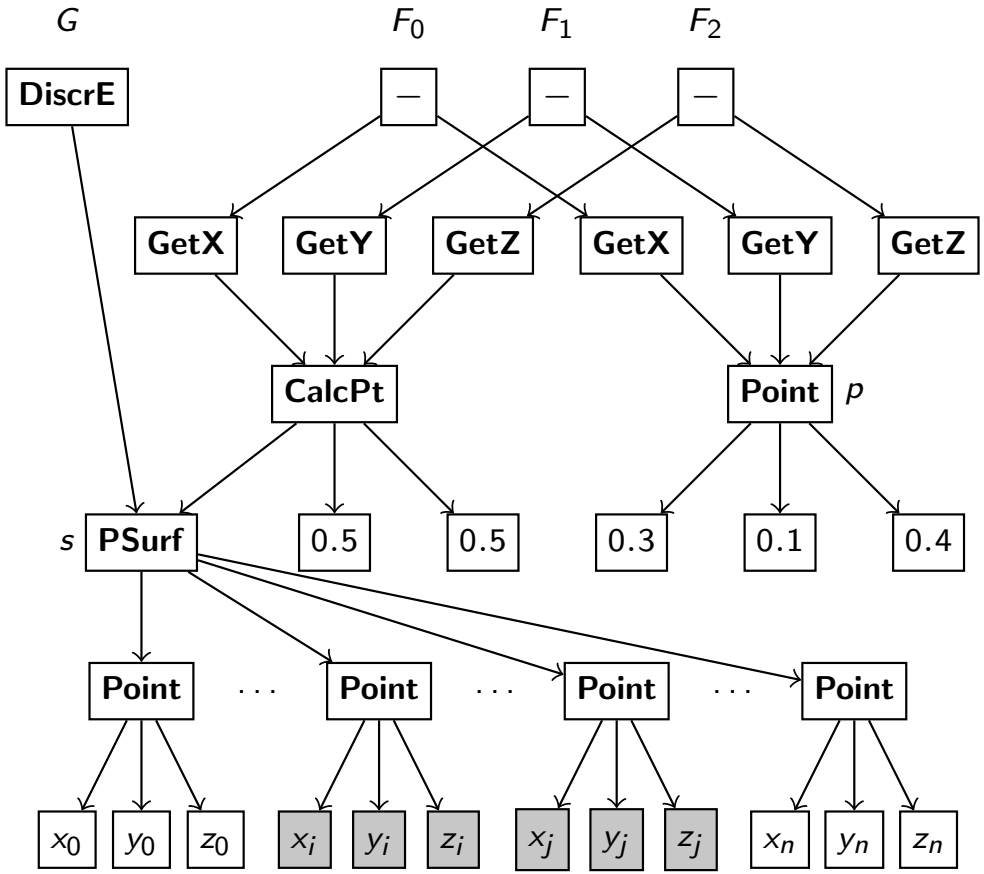
Gouaty et al: Variational geometric modeling with black box constraints and DAGs. CAD 75–76(2016)1–12.



Initial position, final positions: fixed parameters, non fixed parameters.

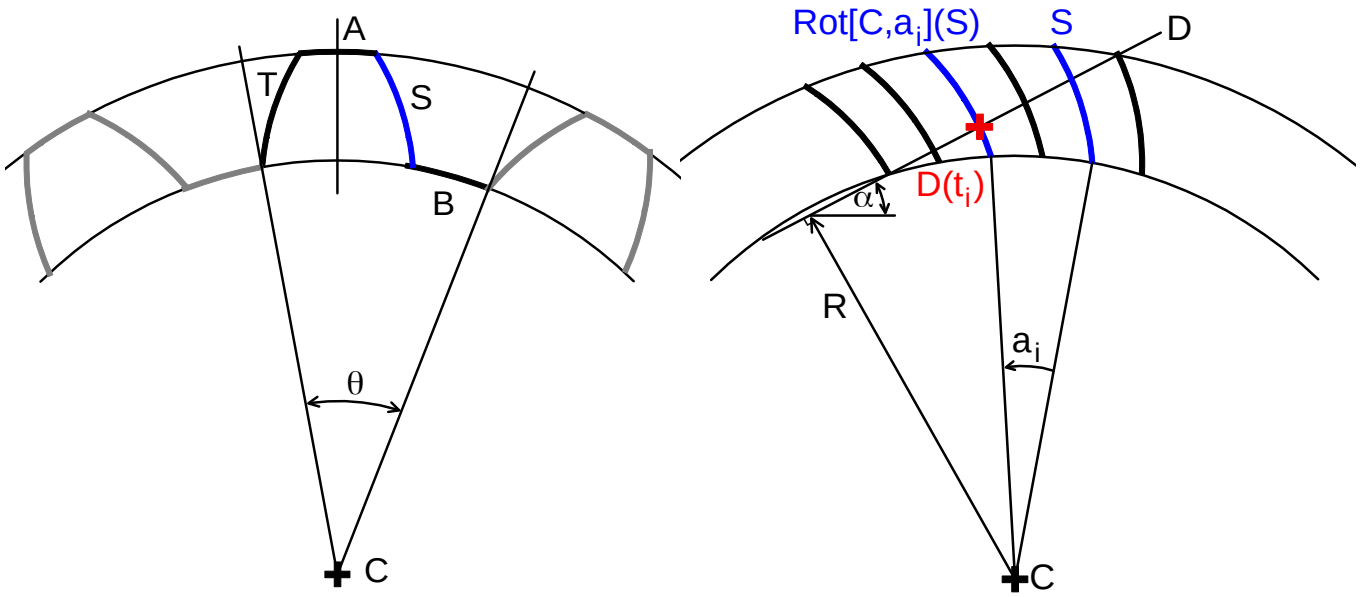
Python source. A given point must lie on a Bspline:

```
1  def point_de_passage() :
2      res = Desc()
3      s = Surf(7, 7)
4      s.bordsFixes()
5      p = Point(0.3, 0.1, 0.4)
6      res.addEqs(Egalite(CalcPt(s, 0.5, 0.5), p))
7      res.fMin = EnDiscr(s)
8      res.addObj(s)
9      return res
10
11 def Egalite(p1, p2) :
12     res = Desc()
13     res.addEq(getX(p1) - getX(p2))
14     res.addEq(getY(p1) - getY(p2))
15     res.addEq(getZ(p1) - getZ(p2))
16     return res
```

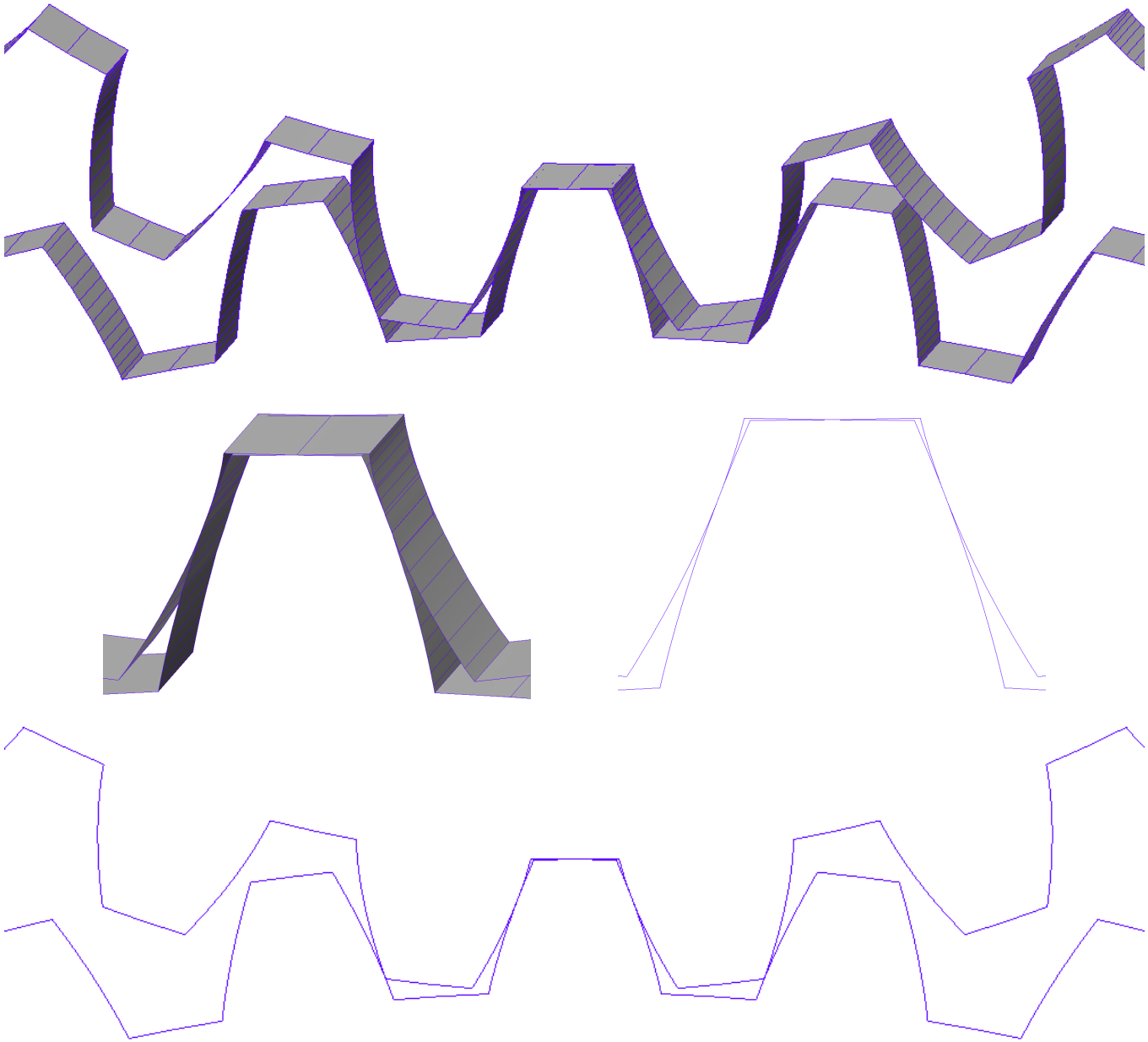


DAG of the system. Unknowns are in gray nodes.

Example 2/3 in CADCAM: 2 gears

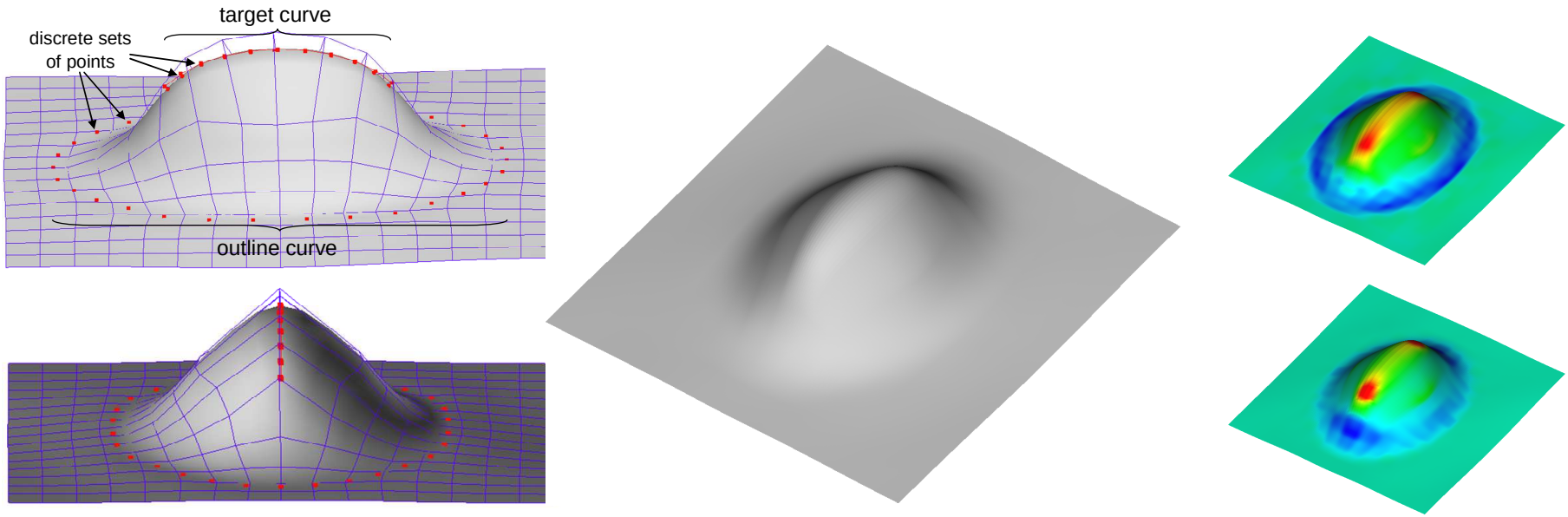


Gears.



The 2 gears (no clearance). 173 equations, 90 unknowns.

Example 3/3 in CADCAM: Hollow in car door



Simulation of an hollow in a car door. Final position satisfying the constraints. Outline and target curves discretization represented by red points (left), visualization of the mean (top right) and Gaussian (bottom right) curvatures. 908 unknowns, 213 equations.

Numerical methods for solving without equations

In CADCAM, a sketch is often available and gives a starting point; the Jacobian is usually very sparse.

- **Newton's method with Jacobian**: it is either approximated with finite differences or exactly computed with dual numbers. For linear algebra, use either sparsity-preserving LUP or Krylov methods (biconjugate gradients) which exploit sparsity.

- **Jacobian-free Newton** method, like the secant method using BFGS formula (google "JFNK")

Numerical methods for solving without equations

- **Homotopy**, starting from the sketch, combined with previous methods

- **minimize** $\sum_i F_i(X)^2$ with **Levenberg-Marquardt, BFGS, Hooke-Jeeves, Nelder-Mead simplex, Torczon simplex, heuristics (Jaya), etc**.

- Sadly, interval methods seem incompatible with black box DAG.

Plan of the talk

The **pentahedron** problem shows the proximity btw Geometric Theorem Proving and Geometric Constraint Solving

The two fields separate: **specificities of GCS**, which goes **from equations to algorithms**.

GCS examples in CADCAM.

GCS still benefits from symbolic tools, like **DAG**, and **dual numbers**.

From algorithms to equations: what if algorithms were just user-friendly way to pose equations, after all?

A common data structure: the DAG

DAG (or Straight Line Program in Dynamic Geometry) is a popular data structure to represent mathematical expressions, or geometric constructions (Cabri, GeoGebra)

In CADCAM, these DAG involve *splines, NURBS, algorithms and algorithmic shapes, rounding or boolean operations (history-based, parametric modelling) $\Rightarrow$ **they are no more (easily) convertible to polynomials, nor differentiable.**

These DAGs have still some interesting features:

Features of DAGs

DAGs can still be evaluated/updated for given numerical values $\Rightarrow$ we can numerically solve

Interactively editable by users $\Rightarrow$ they can pose their problems.

Probabilistic tests (nullity / equality) still possible, up to tolerance.

Exact computation of derivatives is still possible with **dual numbers** – indeed!

Plan of the talk

The **pentahedron** problem shows the proximity btw Geometric Theorem Proving and Geometric Constraint Solving

The two fields separate: **specificities of GCS**, which goes **from equations to algorithms**.

GCS examples in CADCAM.

GCS still benefits from symbolic tools, like **DAG**, and **dual numbers**.

From algorithms to equations: what if algorithms were just user-friendly way to pose equations, after all?

Derivatives with finite differences

To compute derivatives at a given point (convenient for Newton solvers, or BFGS optim), we can use finite differences:

$$\frac{\partial f}{\partial x}(x, Y) \approx (f(x+\epsilon, Y) - f(x-\epsilon, Y))/(2\epsilon) \text{ with } \epsilon = 10^{-5}$$

but they are inaccurate ($\Rightarrow$ convergence pbm close to the optima).

Plan of the talk

The **pentahedron** problem shows the proximity btw Geometric Theorem Proving and Geometric Constraint Solving

The two fields separate: **specificities of GCS**, which goes **from equations to algorithms**.

GCS examples in CADCAM.

GCS still benefits from symbolic tools, like **DAG**, and **dual numbers**.

From algorithms to equations: what if algorithms were just user-friendly way to pose equations, after all?

Derivatives with Dual numbers

Dual numbers: $\epsilon_1, \ldots \epsilon_n$ for $x_1, \ldots x_n$, with $\epsilon_i \epsilon_j = \epsilon_i^2 = 0$

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

Dual numbers

The bijection btwn dual numbers $\leftrightarrow$ matrices is an isomorphism:

The matrice of the opposite is the opposite of the matrice.

The matrice of the inverse is the inverse of the matrice (if invertible).

The matrice of the power is the power of the matrice.

$$\begin{array}{ccccc} (a+b\,\epsilon) & \times & (a'+b'\,\epsilon) & = & aa'+(ab'+ba')\,\epsilon \\ \downarrow & & \downarrow & & \downarrow \\ \begin{pmatrix} a & 0 \\ b & a \end{pmatrix} & \times & \begin{pmatrix} a' & 0 \\ b' & a' \end{pmatrix} & = & \begin{pmatrix} aa' & 0 \\ ba'+ab' & aa' \end{pmatrix} \end{array} \tag{2}$$

Dual numbers

$$\frac{1}{a+b\,\epsilon} = \frac{1}{a} - \frac{b}{a^2}\,\epsilon \text{ when } a \neq 0 \tag{3}$$

Remark: ϵ has no inverse (GB...)

$$(a+b\epsilon)^k = a^k + ka^{k-1}b\,\epsilon \tag{4}$$

$$\begin{aligned} P(x_v+\epsilon) &= a(x_v+\epsilon)^3 + b(x_v+\epsilon)^2 + c(x_v+\epsilon) + d \\ &= a(x_v^3 + 3x_v^2\,\epsilon) + b(x_v^2 + 2x_v\,\epsilon) + c(x_v+\epsilon) + d \\ &= (ax_v^3 + bx_v^2 + cx_v + d) + (3ax_v^2 + 2bx_v + c)\,\epsilon \\ &= P(x_v) + P'(x_v)\,\epsilon \end{aligned} \tag{5}$$

Dual numbers

It extends to multivariate polynomials:

$$\begin{aligned} Q(x_v + \epsilon, y_v) &= Q(x_v, y_v) + Q'_x(x_v, y_v)\epsilon \\ Q(x_v, y_v + \epsilon) &= Q(x_v, y_v) + Q'_y(x_v, y_v)\epsilon \end{aligned} \tag{6}$$

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

Dual numbers

$\mathbb{R} \rightarrow$ quaternions (for representation of rotations in 3D)

$\mathbb{R} + \epsilon\mathbb{R} = \mathbb{R}[\epsilon]/(\epsilon^2 = 0) \rightarrow$ dual quaternions, aka biquaternions (for representation of rotations & translations in 3D)

Many viewpoints of dual numbers

- intuitively, ϵ is an infinitesimal number

- a kind of algebraic extension of $\mathbb{R}$: $\mathbb{R}[\epsilon]/(\epsilon^2 = 0)$ (similarly $\mathbb{C} = \mathbb{R}[i]/(i^2 + 1 = 0)$)

- Taylor expansion up to degree 1

- non standard analysis (Robinson)

- in GB parlance, we exploit the fact that ϵ is not in the ideal of ϵ^2, but in its radical: $\epsilon^2 = 0 \not\Rightarrow \epsilon = 0$.

It extends to transcendental functions:

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

It extends to many ϵs:

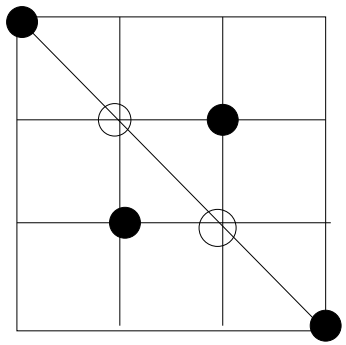
$$\exp(a + b_1\epsilon_1 + b_2\epsilon_2) = e^a + b_1 e^a \epsilon_1 + b_2 e^a \epsilon_2$$

$$F(X_v + \sum_k \epsilon_k) = F(X_v) + \sum_k \partial F/\partial x_k(X_v)\epsilon_k$$

It applies to determinantal equation $|M(X)| = 0$. There is also a formula for $|A + \epsilon B|$:

$\det(I + \epsilon M) = 1 + \text{Trace}(M)\,\epsilon$

$$\begin{aligned} \det(I + \epsilon M) &= (1 + M_{11}\epsilon)(1 + M_{22}\epsilon)\ldots(1 + M_{nn}\epsilon) + \ldots \\ &= 1 + \text{Trace}(M) + \ldots \quad \text{with standard matrice } M \end{aligned}$$



Other matchings use at least 2 off-diagonal entries, thus are 0.

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

$$\begin{aligned} \det(A+\epsilon B) &= \det(A(I+\epsilon A^{-1}B)) \\ &= \det(A)\det(I+\epsilon A^{-1}B) \\ &= \det(A)(1+\mathrm{Trace}(A^{-1}B)\,\epsilon) \end{aligned} \tag{7}$$

If A not inversible: $A = U\Sigma V^t$ (Σ diagonal and U, V unitary):

$$\begin{aligned} \det(A + \epsilon B) &= \det(U\Sigma V^t + \epsilon B) \\ &= \det(U(\Sigma V^t + \epsilon U^t B)) \\ &= \det(U(\Sigma + \epsilon U^t B V)V^t) \\ &= \det(\Sigma + \epsilon U^t B V) \end{aligned} \tag{8}$$

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

$$(\sigma_1 + k_1\epsilon)\dots(\sigma_{n-1} + k_{n-1}\epsilon)(0 + k_n\epsilon) = 0 + \sigma_1 \dots \sigma_{n-1} k_n \epsilon$$

Dual numbers...

provide exact (up to f.p. precision) derivatives $\Rightarrow$ we can use Newton's method to solve, or Euler method to follow an homotopy curve, or BFGS to minimize, etc, **even when equations are not available**.

It is possible to compute Taylor expansions beyond degree 1 (using $\epsilon^5 = 0$), which eases order 4 Runge Kutta method for homotopy. The cost can be mitigated exploiting sparsity of ϵ expansions. For sorting ϵ expansions, we can use GB compatible orders.

Plan of the talk

The **pentahedron** problem shows the proximity btw Geometric Theorem Proving and Geometric Constraint Solving

The two fields separate: **specificities of GCS**, which goes **from equations to algorithms**.

GCS examples in CADCAM.

GCS still benefits from symbolic tools, like **DAG**, and **dual numbers**.

From algorithms to equations: what if algorithms were just user-friendly way to pose equations, after all?

From algorithms to equations

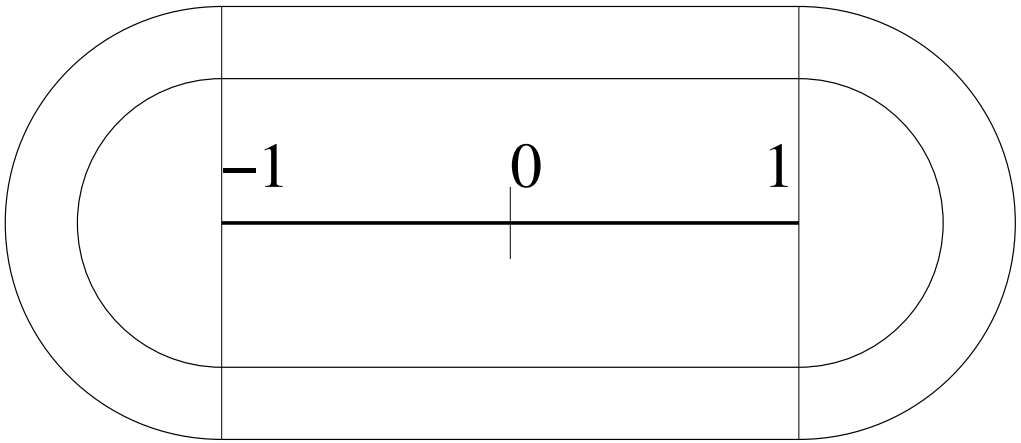
Caution: The most conjectural part of the talk !

We followed GCS which went **from equations to algorithms**.

Now let us go **from algorithms to equations**.

Algorithms, if-then-else, $\min, \max, |.|$ are more convenient than equations, but equivalent equations exist:

Which system of equations for the distance between a point (x, y) and a segment $[(x_1, y_1), (x_2, y_2)]$?



$$
\begin{array}{lcllll}
d(x,y)^2 & = & \min(x+1,0)^2 & +y^2 & +\max(0,x-1)^2 \\
x \leq -1 & \Rightarrow & (x+1)^2 & +y^2 & +0^2 \\
-1 \leq x \leq 1 & \Rightarrow & 0^2 & +y^2 & +0^2 \\
1 \leq x & \Rightarrow & 0^2 & +y^2 & +(x-1)^2
\end{array}
$$

An algebraic system which computes sgn(a)

Let $a \in \mathbb{R}$, and $s = \mathrm{sgn}(a)$: $s = \frac{a}{|a|}$ iff $a \neq 0$ and 0 otherwise.

For given a, the algorithm to compute s and $|a|$ is obvious.

But is there an equivalent system of polynomial equations

$$S(a, s) = 0 \Leftrightarrow s = \mathrm{sgn}(a)$$

If yes, it is possible to translate algorithms (not all) into systems of equations.

An algebraic system which computes sgn(a)

Let $a \in \mathbb{R}$, and $s = \text{sgn}(a)$.

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

$a > 0 \Rightarrow$ only one real solution $y^2 = |a| = a$, $s = 1$, $z = 1/a$.
$a < 0 \Rightarrow$ only one real solution $y^2 = |a| = -a$, $s = -1$, $z = -1/a$.
$a = 0 \Rightarrow s = 0$ and y^2 is free. For uniqueness, add the equation:

$$(1 - s^2)(y - 1) = 0$$

$a \neq 0 \Rightarrow 1 - s^2 = 0 \Rightarrow$ it does not constraint y.
$a = 0 \Rightarrow s = 0, 1 - s^2 = 1 \Rightarrow$ the constraint $y - 1 = 0$ becomes active and the only real solution is $s = 0, y = z = 1$.

From algorithms to equations

Then it is easy to build other systems $S(a, R)$ for:

$$R = |a| = sa$$
$$R = a^+ = \max(0, a) = (a + |a|)/2 = (a + sa)/2$$
$$R = a^- = \min(0, a) = a - \max(0, a) = (a - |a|)/2 = (a - sa)/2$$

and also

$$R = \max(a, b) = (a + b)/2 + |b - a|/2$$
$$R = \min(a, b) = (a + b)/2 - |b - a|/2$$

From algorithms to equations

Now we can translate the instruction:

$R = (\text{if } x > 0 \text{ then } P \text{ else if } x < 0 \text{ then } N \text{ else } Z)$

into equations:

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

$x < 0 \Rightarrow R = N$ (negative N is defined in another system)
$0 < x \Rightarrow R = P$ (positive $P \ldots$)
$0 = x \Rightarrow R = Z$ (zero $Z \ldots$)

Rk: $(1 - (\text{sgn}(x^+) + \text{sgn}(x^-))Z = (1 - (\text{sgn}(x))^2)Z$

From algorithms to equations

Translating arithmetic constraints $x \in \mathbb{Z}$ into equations:
$x \in \mathbb{Z} \Leftrightarrow \sin(\pi x) = 0$. The system has finite size and defines all integers in $\mathbb{Z}$. But it is not algebraic.

Algebraic system: for $x \in [0, 2^n - 1]$, $x = x_0 + 2x_1 + \ldots 2^n x_n$ and $x_i(1 - x_i) = 0$, $\forall i \in [0, n]$. Log size. But it does not define all integers.

Remark: $x(x-1)(x-2)\ldots(x-2^n+1) = 0$ is not good because of exponential size.

Translating fixpoints into systems of equations

After pure functional languages, we know that assignments and iterations are useless.

To compute fixpoints: $F(F(...(X)))$:

Assume the program $Y = F(X)$ is represented by a system: $S(X, Y) = 0$

Then fixpoints: $F(F(...(X)))$ are represented by the system $S(X, X) = 0$

Warning! $S(X, X) = 0$ clearly exposes that size(X) = size(Y). Untrue for subdivision curves where size(Y)=2$\times$size(X).

From algorithms to equations

Assume that the conversion: algorithms $\rightarrow$ equations is possible u.m.a.

Easily automatizable.

So what?

GB (say) tools apply to resulting equations, thus to algorithms, and PP (*splines). Interesting!

Sometimes equations are missing but there is an algorithm (to cancel spurious roots), thus we have equations after all!

From algorithms to equations

Maybe algorithms are just a friendly-user way to pose equations?

Maybe a GB of resulting equations provides a better algorithm?

Maybe the GB or primary decomposition characterizes conditions for the algorithm to work/fail ?

Rk: Resulting equations are irrelevant for numeric solvers: inaccuracy, which values for auxiliary variables, convergence issues, etc.

Conclusion

The **pentahedron** problem shows the proximity btw Geometric Theorem Proving and Geometric Constraint Solving

The two fields separate: **specificities of GCS**, which goes **from equations to algorithms**.

GCS examples in CADCAM.

GCS still benefits from symbolic tools, like **DAG**, and **dual numbers**.

From algorithms to equations: what if algorithms were just user-friendly way to pose equations, after all?

Thanks. Questions?