Proceedings of the International Association for
Shell and Spatial Structures (IASS) Symposium 2013
"BEYOND THE LIMITS OF MAN"
23-27 September, Wroclaw University of Technology, Poland
J.B. Obrębski and R. Tarczewski (eds.)

Constraint-Based Graphic Statics: New paradigms of computer-aided structural equilibrium design

Corentin Fivet[1], **Denis Zastavni**[2]

[1]PhD candidate, Faculty of Architecture, Architectural Engineering, Urban Planning, UCLouvain, Belgium, *corentin.fivet@uclouvain.be*
[2] Associate Prof., Faculty of Architecture, Architectural Engineering, Urban Planning, UCLouvain, Belgium, *denis.zastavni@uclouvain.be*

Summary: This paper presents a new computer-aided environment aimed at assisting designers during the very first stages of the structural design process. It gives designers the opportunity to build and modify interactively geometric constraints that control the structural shape and its static equilibrium simultaneously and in an entirely graphical way. Following a brief contextualisation, the paper presents the main features of the inner algorithms. Typical uses are then described in order to illustrate the wide range of applications and their practical benefits. As a result, it is argued that this environment opens up new opportunities for precognitive, chronology-free and highly interactive paradigms of structural equilibrium design.

Keywords: *structural design; static equilibrium; graphic statics; constraint-based geometry; computer-aided design*

1. INTRODUCTION

1.1. Structural morphogenesis

Which geometric changes would make a particular rod in a truss become in tension? Which additional weight would enable the reaction stresses on a given footing to be sufficiently vertical so as to be supported by the ground? Which cross-sectional shape would enable the bending moments of a certain free-form beam to be at a constant desired value? Which inertia …? Which slope …? Which load …?

These are important structural design issues [1], yet they still cannot be resolved without time-consuming empirical iterations. Past structural design practices [2, 3] have shown that if these issues are resolved quickly and easily, it then allows the structural behaviour to be clearly stated from the outset — *i.e.* when the first choices about the structural shape and its equilibrium are made, ahead of any subsequent structural analysis. This stage is known as structural morphogenesis.

Owing to the fact that it is usually harder to amend these initial choices at a later, more detailed stage of the design process, structural morphogenesis is crucially important to ensuring a structural shape that is robust and consistent with economic and ecological demands.

1.2. Proposal and objectives

This paper presents a new computer-aided environment aimed at facilitating and improving the interactive definition of structural behaviours. This environment is built on the following hypotheses:

– since load paths condition most structural behaviours, this environment allows the user to create, modify, combine and refine any strut-and-tie network with operations that are geometrically channelled so that it remains in static equilibrium at each step of the process;

– since all structural behaviours depend on the structural shape and its inner stresses simultaneously, this environment takes full advantage of the visual expressiveness of graphic statics [4, 5];

– since the user ought to maintain full and permanent control over all the choices, the computer's unique role is to inform the user about the consequences of these choices and hence the remaining possibilities. The proposed environment meets this objective by applying a graphical region — *i.e.* a solution domain — to each point of both reciprocal diagrams of the strut-and-tie network (see point p* on Fig. 1).

Although these concepts might somehow be extended to the third dimension, the environment proposed in this paper deals solely with planar equilibriums — which does not impede the design of most spatial structures.

Section 2 presents the features of the environment. Section 3 introduces various uses of this environment and serves as an illustration of the discussion in Section 4. However, before that, the next sub-section compares this environment with other existing tools.

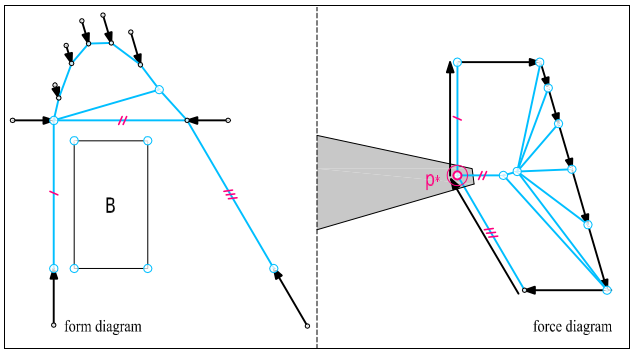

Fig. 1: As long as point p* remains inside the filled area, reciprocal adjacent rods in the form diagram never cross the rectangle B — *i.e.* a geometric condition B borders force amplitudes graphically. See Fig. 15 page 5 for practical use of this strut-and-tie network.

1.3. State of the art

This environment can be compared with others that share part of its nature: computer-aided strut-and-tie modellers, dynamic geometry software and computer-aided graphic statics.

– Nearly all current computer-aided strut-and-tie modellers — whether as parts of multi-purpose structural analysis tools [6, 7] or plugins of geometric modellers [8, 9] — fail to support a chronology-free design process, *i.e.* they either compute the inner forces from a predetermined shape or the other way round, but they give no opportunity to reverse the deductive approach as it goes along. The exceptions to this rule — *e.g.* [10] — consider very specific structural typologies.

– To the best of the author's knowledge, current dynamic geometry software [11, 12] offer little support for constructions allowing multiple solutions, graphical inequalities, relative directions, interdependent constraints and non-destructive switching of the dependencies hierarchy. These very useful concepts are presented further in the paper.

– Classical graphic statics — *i.e.* successive drawings on a sheet of paper [4, 5] — has the disadvantage of constructing the reciprocal diagrams in a process that can soon become slow and tedious since the equilibrium is only obtained at the very end of the process. Some computer-aided implementations of graphic statics have been developed [13, 14, 15]. They allow a dynamic displacement of nodes on pre-assembled structures but do not allow their interactive construction.

2. METHODOLOGY

2.1. General overview

The main methodological approach is to describe every variable, equation and object completely geometrically. It follows that the only

parameters are positions of points. On the one hand, this feature has no limitations since angles, lengths, alignments, proportionalities *etc.* can actually all be defined with positions of points. On the other hand, this homogeneous treatment considerably simplifies the core algorithms.

Starting from scratch, the designer has a set of instructions available that allow any strut-and-tie model to be built, modified and constrained stage by stage. These instructions are defined as linear procedures calling a small set of primitive operations.

The next sub-sections identify the entire set of primitive operations: sub-section 2.2 presents primitive operations enabling the application and cancellation of powerful geometric constraints; particular constraints are illustrated in sub-sections 2.3 and 2.4; sub-section 2.5 introduces a non-destructive technique to switch the hierarchy of dependencies of constraints; lastly sub-sections 2.6 and 2.7 define the necessary primitive operations to modify strut-and-tie networks and express the properties they must hold to ensure static equilibrium.

2.2. Constructing geometric inequalities and directionalities

The very first primitive operation merges two points if they are coincident. The next primitive operations combine, apply and cancel geometric constraints. These geometric constraints are planar graphical regions that result from a Boolean combination — either unions "∪", intersections "∩" or negations "\" — of three fundamental geometric constraints: HalfPlane, DiscInside and DiscOutside.

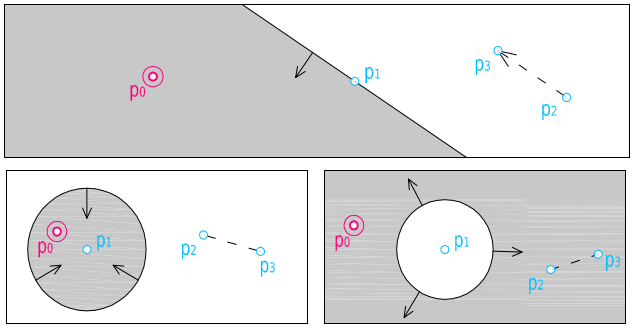

Fig. 2: (up) a point p_0 that belongs to a HalfPlane$[p_1p_2p_3]$ constraint; (bot. left) a point p_0 that belongs to a DiscInside$[p_1p_2p_3]$ constraint; (bot. right) a point p_0 that belongs to a DiscOutside$[p_1p_2p_3]$ constraint.

Each of these three constraints takes three points as parameters. They are defined as follows:

– a point p_0 belongs to a constraint HalfPlane$[p_1\ p_2\ p_3]$ if it is positioned on the left of or in line with p_1 according to the direction from p_2 to p_3 when p_2 and p_3 are not coincident — the corresponding graphical region is a closed half-plane (Fig. 2, up); when p_2 and p_3 are coincident, the corresponding region is the entire plane, meaning that the membership is always verified

– a point p_0 belongs to a constraint DiscInside$[p_1\ p_2\ p_3]$ if the distance from p_0 to p_1 is less than or equal to the distance from p_2 to p_3 — the corresponding region is a closed disc (Fig. 2, bottom left)

– a point p_0 belongs to a constraint DiscOutside$[p_1\ p_2\ p_3]$ if the distance from p_0 to p_1 is greater than or equal to the distance from p_2 to p_3 — the corresponding region is the inverse of an open disc (Fig. 2, bot. right).

These constraints are sufficient to build any compass-and-straightedge construction, including graphical arithmetic and trigonometry. The following definitions merely illustrate the creation of non-fundamental constraints that can be used as a line or a circle respectively:

Straightedge$[p_0p_1p_2]$:= HalfPlane$[p_0p_1p_2]$ ∩ HalfPlane$[p_0p_2p_1]$
Compass$[p_0p_1p_2]$:= DiscInside$[p_0p_1p_2]$ ∩ DiscOutside$[p_0p_1p_2]$

For another example, the OrthoProj$[p_0p_1p_2]$ constraint is defined as follows (Fig. 3):

OrthoProj$[p_0p_1p_2]$:= Straightedge$[p_0p_0p_1]$ ∩ Straightedge$[p_2p_3p_4]$ *where:*
$p_3 \in$ Compass$[p_0p_0p_1]$ ∩ Compass$[p_1p_0p_1]$ ∩ HalfPlane$[p_0p_0p_1]$
$p_4 \in$ Compass$[p_0p_0p_1]$ ∩ Compass$[p_1p_0p_1]$ ∩ HalfPlane$[p_0p_1p_0]$

Applying this constraint on a point p_5 — *i.e.* "$p_5 \in$ OrthoProj$[p_0p_1p_2]$" — means that p_5 is constrained on the orthogonal projection of p_2 onto a line passing through p_0 and p_1. This example emphasises the variational quality of the approach — *i.e.* the resulting geometric statement is always identical irrespective of the order of the application of constraints.

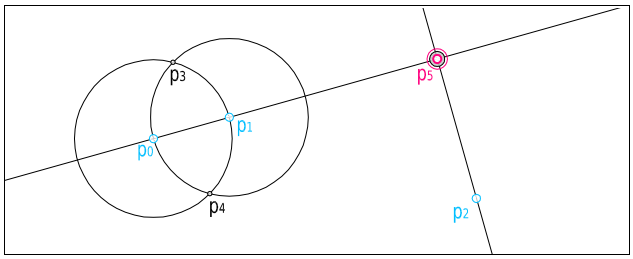

Fig. 3: Detailed construction of a OrthoProj$[p_0p_1p_2]$ constraint and application of this constraint to p_5.

However the three fundamental constraints can achieve many more applications than the compass and straightedge since (1) they can be combined with unions and negations, (2) they depict inequalities (instead of equalities) and (3) the HalfPlane constraint incorporates a notion of relative directionality. The following construction illustrates these capabilities (Fig. 4):

$p_5 \in$ (\ (HalfPlane$[p_0p_0p_1]$ ∪ HalfPlane$[p_0p_2p_0]$) ∪ HalfPlane$[p_0p_3p_4]$)
∩ DiscOutside$[p_0p_0p_1]$ ∩ DiscInside$[p_0p_0p_2]$

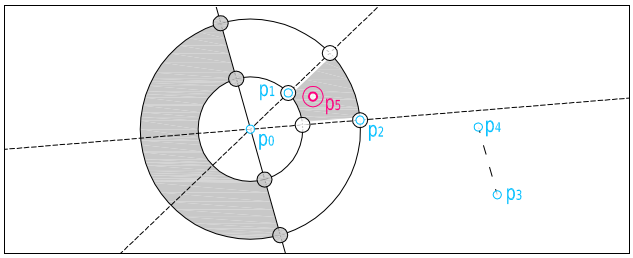

Fig. 4: Application onto p_5 of a Boolean combination of constraints using the concepts of inequality and relative directionality.

In this last example, moving p_1 just beyond p_5 would cause p_5 to be outside its own domain of solutions. In order to prevent this, the environment automatically moves p_5 to its closest position inside its domain. This algorithm — the only one that has to be performed so far when points are moving — is very efficient since the closest position is either an orthogonal projection onto a line or a circle, or an intersection of lines and circles. For another example, the same automation in Fig. 3 ensures that moving p_2 will update the position of p_5.

In the example of Fig. 4, moving p_1 beyond p_2 would cause the domain of solutions of p_5 to be empty, which means that the geometric statement cannot be solved. In order to prevent all the parameters that constrain p_5 from emptying its domain — *i.e.* to ensure arc consistency [16] —, the environment automatically propagates constraints to these points. For example, the automatic constraint that should be applied to p_1 is DiscInside$[p_0p_0p_2]$. For most cases, this mechanism presents the great advantage of being executed symbolically — *i.e.* the application is only computed once when the user alters the construction plan, and does not have to be performed at each point displacement.

2.3. Moving points on special curves using interdependencies

Interdependency is created when a certain point is constrained by itself, directly or through in-between parameters. Interdependency is of great importance because it offers an elegant method for constraining points

on any algebraic curve, using only the three aforementioned fundamental geometric constraints. The sole additional mechanism that has to be implemented in the environment is to stop the iterative automatic displacement onto the closest position when it converges.

As an illustration, a point p_6 is here constrained on a hyperbola (Fig. 5, inspired by [17] page 183). Moving p_6 will update the position of p_5 and consequently the position of p_6, until it converges on the hyperbola.

$p_4 \in$ Straightedge$[p_0p_0p_6] \cap$ Compass$[p_0p_2p_3]$
$p_5 \in$ Compass$[p_4p_0p_1] \cap$ Compass$[p_1p_2p_3] \cap$\Straightedge$[p_1p_0p_4]$
$p_6 \in$ Straightedge$[p_1p_1p_5]$

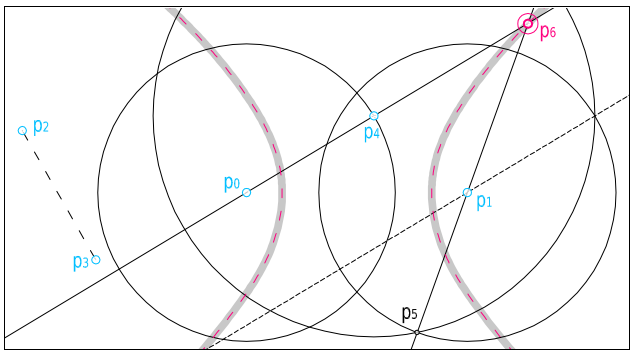

Fig. 5: Detailed construction that forces p_6 to remain on a hyperbola.

2.4. Applying dynamic conditional statements

Dynamic conditional statements are simple instructions that apply particular constraints whose graphical region is either a unique position or its inverse — *i.e.* the entire plane minus this position — depending on whether a Boolean geometric condition is currently true or false.

The definition of these conditional constraints being beyond the scope of this paper, the next example simply shows the basic capability of defining a constraint that returns the greater of two distances:

MaxDistance$[p_0p_1p_2p_3p_4]$:= Compass$[p_0p_0p_7]$ *where:*
$p_5 \in$ Compass$[p_0p_1p_2]$
$p_6 \in$ Compass$[p_0p_3p_4]$
$p_7 \in$ DiscOutside$[p_0p_0p_5] \cap$ DiscOutside$[p_0p_0p_6]$
$\cap$ (DiscInside$[p_5p_5p_5] \cup$ DiscInside$[p_6p_6p_6]$)

If MaxDistance$[p_0p_1p_2p_3p_4]$ is applied to a point p_8, then the distance p_0p_8 will always be the greatest of the two distances p_1p_2 and p_3p_4, whatever they are or are becoming (Fig. 6). As a result, powerful dynamic algorithms can be created in an entirely graphical way.

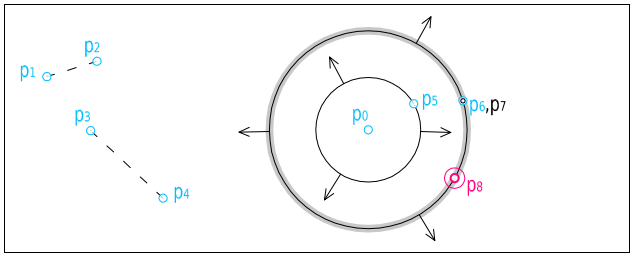

Fig. 6: Whatever these positions are or are becoming, the distance p_0p_8 will always be the greatest of the two distances p_1p_2 and p_3p_4.

2.5. Switching constraint dependencies

The following primitive operation allows the user to switch the hierarchy between a point and its parameters without having to rebuild the construction. The unprecedented consequence of this feature is that the user does not have to predict the parametric hierarchy in advance or rebuild the model whenever it has to be changed. The solution of a problem can become the statement of the inverse problem and vice versa.

This feature actually benefits from the symmetric properties of the three fundamental constraints — *e.g.* constraining a point p_0 on HalfPlane$[p_1p_2p_3]$ has the same geometric consequence as constraining p_1 on HalfPlane$[p_0p_3p_2]$, p_2 on HalfPlane$[p_3p_1p_0]$ or p_3 on HalfPlane$[p_2p_0p_1]$.

In order to illustrate the point, the example of Fig. 3 is reused. In this construction, p_5 is dependent of p_2 and cannot be moved away from the intersection of two straight edges. Because the application of Straightedge$[p_2p_3p_4]$ on p_5 is symmetric to the application of Straightedge$[p_5p_3p_4]$ on p_2, this direct dependency can be inverted. As a result, p_5 gains one degree of freedom — *i.e.* it can now move along the line passing through p_0 and p_1 — and p_2 loses one degree of freedom — *i.e.* it must now stay on a perpendicular line passing through p_5.

This method remains valid with more complex constraints involving indirect dependencies — *e.g.* when a great-grandfather is asked to become the great-grandchild of its own great-grandchild. The environment switches the dependencies automatically as soon as the two points given by the user are relatives.

2.6. Constructing static equilibria

The following two sub-sections define primitive operations that handle forces, struts and ties within the strong geometric framework established in the previous sub-sections.

Forces are here defined exclusively geometrically by four positions of points within two diagrams: a form diagram in which a distance between two points is measured in units of length and a force diagram in which a distance between two points is measured in Newtons. Concretely, "$p_0 \in$ Force$[p_1p_2p_3]$" means that a force exists that (1) is applied on p_1 in the form diagram, (2) has an intensity equal to the distance p_2p_3, (3) has a direction going from p_2 to p_3 and (4) applies a push on p_1 if directions p_0p_1 and p_2p_3 are equal or applies a pull on p_1 if not (see Fig. 7).

Three new primitive operations are needed to handle balanced forces. The first two create and remove a zero force, defined as $p_0 \in$ Force$[p_1p_2p_2]$. The third resolves a given force into two components — *e.g.* $p_0 \in$ Force$[p_1p_2p_3]$ (Fig. 7) becomes $p_4 \in$ Force$[p_1p_2p_5]$ and $p_6 \in$ Force$[p_1p_5p_3]$ (Fig. 8). The opposite operation that produces the resultant of the two components is obtained by moving p_5 on p_2 (Fig. 8) and then merging them with the first primitive operation (see sub-section 2.2).

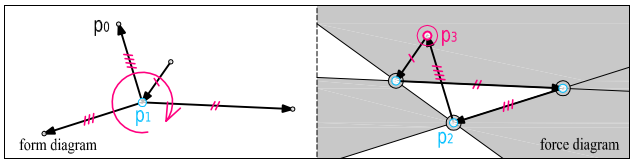

Fig. 7: Four forces balanced on a point. The filled region represents the domain that p_3 must hold to maintain a constant reading cycle around p_1.

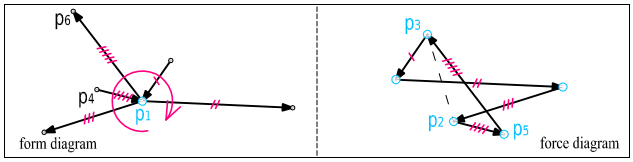

Fig. 8: The same balanced point after resolving the force whose magnitude was the distance p_2p_3.

The successive application of these last primitive operations not only allows the interactive definition of any set of forces applied on a point in the form diagram, but also guarantees that this set of forces stays in static equilibrium — *i.e.* that the corresponding force polygon in the force diagram remains closed.

In order for both diagrams to be reciprocal, each force polygon must be read in an order that is identical to the one in which forces are encountered around the corresponding point of application (in the form diagram), according to a direction that is always either clockwise or anti-clockwise. The proposed environment satisfies this condition by

automatically applying adequate geometric constraints on points in the force diagram (*e.g.* the domain of p_3 in Fig. 7).

2.7. Constructing strut-and-tie networks

The final two primitive operations create and cancel a rod. Rods have the particularity of not being defined as a new object. They are simply defined as two forces meeting specific geometric properties: they must be of equal amplitude, equal orientation, opposite direction and aligned in the form diagram (Fig. 9).

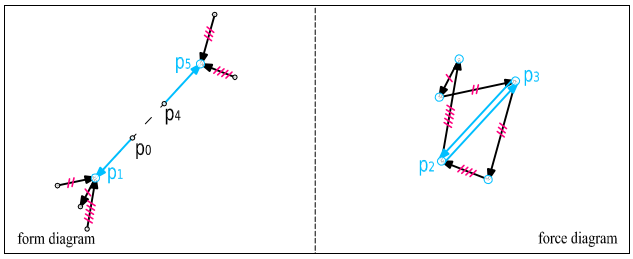

Fig. 9: A pair of forces that is ready to be transformed into a rod.

Again, this definition guarantees the conservation of the reading cycle of forces in both reciprocal diagrams (Fig. 9). However, the consequence of reciprocity is that it prevents a rod in the form diagram from being crossed with either another rod or the half-line behind a force. The proposed environment systematically manages this drawback by dividing such rods into two parts (such a case is shown by p_4 on Fig.17).

2.8. Constraining strut-and-tie networks

The small set of primitive operations that has been identified in this section can easily be placed one after the other in order to define new operations that perform more complex purposes. Such routines can be used to create standard networks — *e.g.* catenaries or basic trusses — or to apply typological or boundary conditions, as will be seen in section 3.

3. APPLICATIONS

3.1. General overview

The strut-and-tie networks generated can be used as general abstractions of the stability of a very wide range of structures: isostatic or hyperstatic reticular structures, mechanisms, self-stressed structures, prestressed structures, irregular beams subjected to bending, compression-only structures, discontinuous stress fields, soil mechanics *etc.* Most of these applications take full benefit of the safe theorem of plasticity [18].

The following exemplifications attempt to illustrate the various ways of working permitted by this environment.

3.2. Constructing and optimising a reticular structure

Question: Which shape should be adequate for a given distributed load? This structural issue has an infinite number of solutions. The ultimate solution will be the result of all the successive choices made by the user during the interactive construction of the strut-and-tie model.

Fig. 10 shows the initial settings: a balanced set of distributed loads. The role of the designer is to focus on what operations should be applied to ensure that the only remaining forces are the loads or the reactions at the supports. If the structure has to be self-stressed, the goal would be to transform all the forces into struts or ties. Fig. 11, Fig. 12 and Fig. 13 summarize a possible process.

The point p** (Fig. 13) is then moved to make the rod r_0 a strut, initial loads can be refined — which does not require the manual reconstruction of all the operations —, and HalfPlane constraints are added to the form diagram in order to guarantee a sufficiently large usable area below the portico (figure Fig. 1). These constraints are then automatically propagated on every parent point such as p* (see filled area of figure Fig. 1).

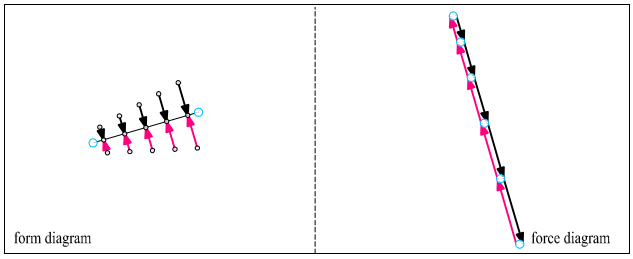

Fig. 10: Initial balanced loads.

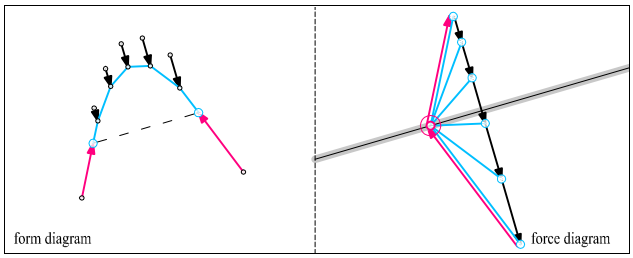

Fig. 11: Application of a catenary arch.

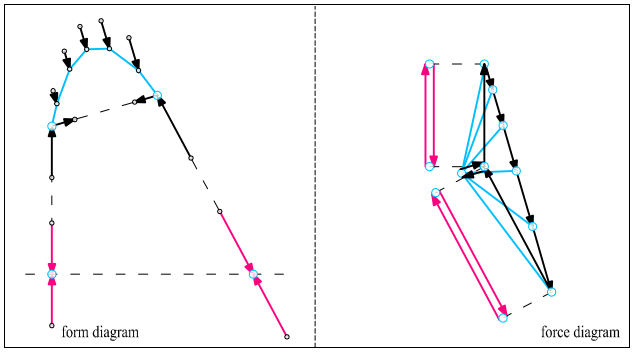

Fig. 12: The reactions of the catenary are resolved and two new balanced points are created.

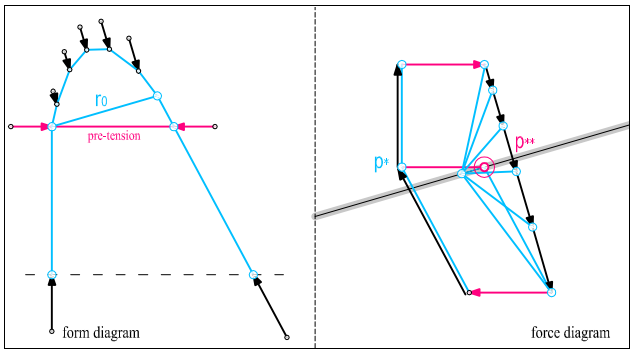

Fig. 13: Two pairs of compatible forces are transformed into two rods and a pre-tensioned tie is added.

Q. Which geometry minimises the forces inside that typology? Since the graphical region associated to every point describes the entire solution domain of the structural issue, they can be used to perform basic optimisations manually. For example, moving p* to the far right of its domain (see Fig. 1) minimises the inner forces of the convergent rods.

3.3. Sketching discontinuous stress fields in plastic materials

Q. Which sketch of reinforcement ties would be needed to withstand the loads applied on a given reinforced concrete shear wall? Thanks to the safe theorem of plastic theory [19], similar interactive form-finding

processes to build discontinuous rectilinear stress fields exist — *i.e.* compound objects of half-planes and rods — (Fig. 14 and Fig. 15, left).

If the geometry of each nodal fields is further constrained to present uniform stress distributions — which is not the case in Fig. 15, left) —, these nodal fields can be studied with Mohr circles in the force diagram.

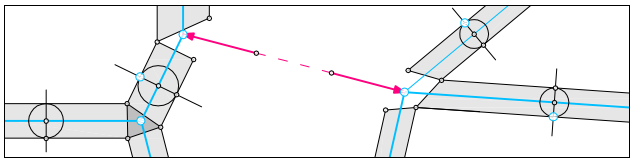

Fig. 14: Construction of a rectilinear discontinuous stress fields using geometric constraints and rods.

3.4. Designing with bending moments

When structural applications follow classical beam theory, the interactive construction of reciprocal diagrams allow the centroids — whether providing or provided by the cross-sections of a beam — and the moment distributions — whether providing or provided by the balanced strut-and-tie network — to be controlled simultaneously.

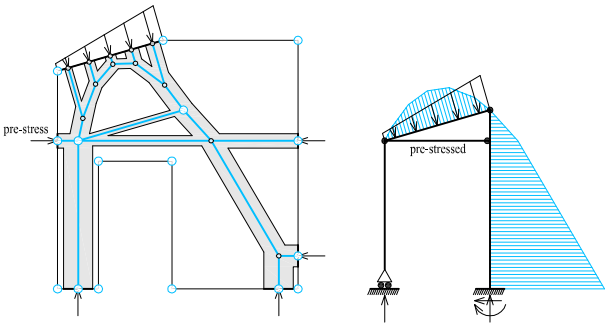

Fig. 15: Left, discontinuous stress fields inside a concrete shear wall; Right, bending moments diagram of the resulting portico.

Q. Given the result of Fig. 1, what is the moment distribution for orientating both columns of the portico vertically? Bending moments (Fig. 15, right) are simply measured by adding together the products of the eccentricities between the struts and the centroids (in the form diagram) and the corresponding force components (in the force diagram) that are parallel to the centroids. Adequate supports can then be designed correspondingly.

Q. What is the minimum bending value for which a clamped beam p_0p_1 must be sized? Using plastic theory, the beam of Fig. 16 can be sized while allowing a hinge somewhere on a point p_9. The moments are here given using a catenary that has to pass through p_1 and p_2 and whose moving pole is p*. The smallest bending moments for which this beam has to be designed are consequently those occurring when the moment on p_0 is equivalent to the one on p_5 or p_6. This configuration is obtained recursively by simply applying on p_2 a compass constraint that is centred in p_0 and whose radius is equal to the greatest distance between p_3p_5 and p_4p_6 (see sub-section 2.4). The position of p_2 and the domain of p* will be then automatically readjusted at each movement of p*.

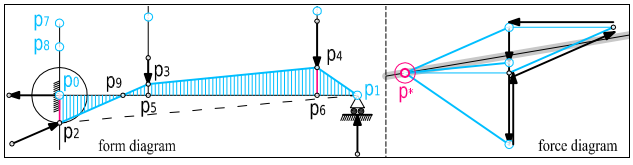

Fig. 16: A clamped beam with possible moment distribution.

3.5. Controlling the structural indeterminacy graphically

Structural indeterminacy can be defined as the ability of a strut-and-tie network to present different inner equilibria under constant loads, meaning that multiple force diagrams can describe the same form diagram. This is encountered here when a point can be moved in the force diagram without changing the orientation of any rod or force.

The frame of figure Fig. 17 describes a simple structure of this kind. It has been built with the following geometric properties:

$p_4 \in$ Straightedge$[p_0p_0p_2] \cap$ Straightedge$[p_1p_1p_3]$
$p_8 \in$ Straightedge$[p_{12}p_2p_3]$
$p_9 \in$ Straightedge$[p_5p_0p_3]$
$p_{10} \in$ Straightedge$[p_9p_0p_4] \cap$ Straightedge$[p_6p_0p_1]$
$p_{11} \in$ Straightedge$[p_{10}p_1p_4] \cap$ Straightedge$[p_7p_1p_2]$
$p_{12} \in$ Straightedge$[p_{11}p_2p_4] \cap$ Straightedge$[p_9p_3p_4]$

Moving p_9 on its own domain — *i.e.* on a line — changes intensities without modifying orientations (Fig. 17, Fig. 18 and Fig. 19). It follows that the domain of p_9 is a comprehensive graphical representation of the indeterminacy of the frame.

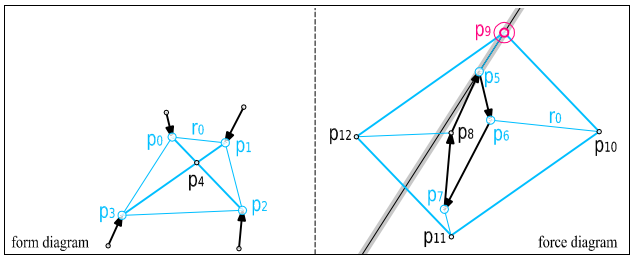

Fig. 17: Possible force distribution inside an indeterminate frame. The domain of p_9 is a representation of its indeterminacy.

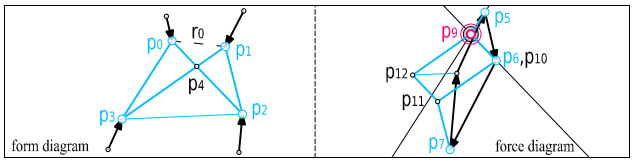

Fig. 18: Another possible force distribution inside the indeterminate frame, in which rod r_0 is superfluous.

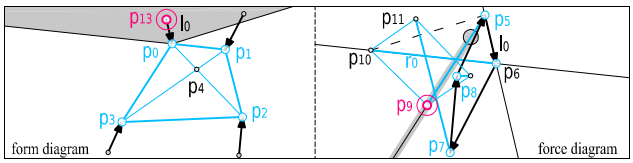

Fig. 19: Another possible force distribution inside the indeterminate frame. The domain of p_{13} reflects all the orientations l_0 can have so that r_0 remains compressed.

Q. Which distribution of forces would make the rod r_0 unnecessary? To cancel the amplitudes of r_0, a geometric constraint can be applied on p_{10} so that p_6 and p_{10} — *i.e.* the ends of the amplitudes of r_0 in the force diagram — are coincident. Thanks to the automatic propagation of constraints (see last paragraph of sub-section 2.2), this has the effect of constraining p_9 on the intersection Straightedge $[p_5p_0p_3] \cap$ Sraightedge$[p_{10}p_0p_4]$, removing the indeterminacy (Fig. 18).

Q. What inertia would make rod r_0 become compressed? From the geometric state of Fig. 18, the domain of p_9 for which r_0 is compressed can be deduced (Fig. 19). In a design context, cross-sections can then be deduced so that the inertias impose the desired stress distribution.

Q. Without changing the stresses inside the inner ties, which orientation of the load l_0 would ensure rod r_0 remains compressed? The first step to solving this issue is to fix the inner tension forces by switching the dependencies between p_6 and p_8 so that p_8 becomes completely free of

constraint and p_6 becomes constrained on Straightedge[$p_{10}p_0p_1$]. Point p_6 must then be further constrained on Straightedge[$p_5p_0p_{13}$] so that the orientation of l_0 is given by point p_{13} in the form diagram. The resulting propagated domain of p_{13} ultimately describes all the orientations that load l_0 can have in order to ensure the compression of rod r_0 (form diagram of Fig. 19).

3.6. Analysing the stability of compression-only structures

The final example is analytical rather than design-oriented: the stability of a given geometry of compressed voussoirs has to be checked. This means that the thrust line balanced with the dead loads of the voussoirs must exist inside the arch geometry [20].

Q. How can the set of possible thrust lines inside the arch of Fig. 20, be characterised fully? The first stage here is to build a set of HalfPlane constraints that corresponds to the geometry of the arch and to apply it to each node of the generated thrust line. These constraints are then automatically propagated on every point that controls the geometry of the thrust line. As a consequence, the near-triangular domain of p_0 (Fig. 20), for example, covers all the domain of stability of the arch. The user knows the minimum and maximum intensities graphically even before moving p_0. And moving p_0 allows each thrust line to be visualised.

If, however, dependencies are switched between p_0 and p_1, the domain of p_1 — a linear segment — will in turn reflect the set of possible thrust lines, this time in the form diagram (Fig. 20, left).

Q. Which minimum value of the weight w* would ensure the stability of the arch? Since point p_2 is also a parameter of the thrust line, its domain reflects the set of all possible thrust lines too (Fig. 20, right). In other words, its graphical domain synthesises all the values of the weight w*, and in particular its minimum value, for which the stability of the arch is guaranteed — for which the thrust line remains enclosed in the masonry.

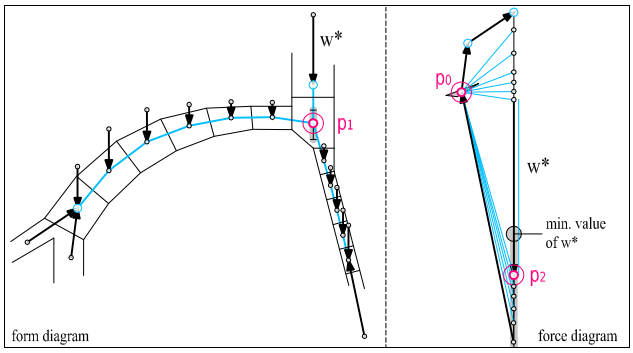

Fig. 20: A masonry arch and its inner thrust line. Highlighted domains represent the domain of stability of this arch.

4. CONCLUSION

This paper has described the features and practical benefits of a new computer-aided environment intended to facilitate and improve the interactive definition of static equilibriums.

Section 1 clarified the importance of structural morphogenesis in the shaping of robust structures that are consistent with economic and ecological demands. It also outlined the restrictions of the proposed environment and compared it with other tools.

Section 2 then documented the various unprecedented features made possible by this environment, such as the ability to:

- handle multiple solutions to a problem at the same time
- automate the definition of graphical regions that reflect the domain of solutions of the current geometric/structural statement
- switch the parametric hierarchy on demand
- compute complete interdependency in order to build algebraic curves
- execute dynamic conditional statements geometrically
- guarantee static equilibrium using only geometric conditions
- build reciprocal diagrams interactively.

Section 3 illustrated how the structural designer can use this environment to produce pre-stressed or hyperstatic reticular structures, beams subjected to bending, compression-only structures and discontinuous stress fields in plastic materials. These applications have demonstrated new uses, such as the ability to:

- build any strut-and-tie network in equilibrium intuitively while controlling structural parts and inner stresses simultaneously
- refine and combine any structural equilibrium incrementally
- perform manual structural optimisations with respect to boundary conditions, using the graphical regions of solutions
- handle structural indeterminacy using graphical regions of solutions.

As a result, this environment opens up new paradigms for *highly-interactive*, *pre-cognitive* and *chronology-free* design exploration: (1) *highly-interactive* because of the user's ongoing control over the structure being shaped and because of the visual expressiveness given by computer-aided graphic statics; (2) *pre-cognitive* because every graphical region of solutions marks out the range of possibilities before they are even explored; and (3) *chronology-free* because the environment allows the control of the geometries, its inner stresses, its static indeterminacy, its boundary conditions and its material strengths independently and simultaneously.

5. REFERENCES

[1] Zastavni D., *An equilibrium approach on a structural scale to structural design*, in: P. Cruz (ed.) *International Conference on Structures and Architecture*, Guimarães, Portugal (2010).

[2] Zastavni D., *The structural design of Maillart's Chiasso Shed (1924): a graphic procedure*, Structural Engineering International, Vol. 18, No. 3 (2008) pp. 247-252.

[3] Fivet, C., Zastavni D., *The Salginatobel bridge design process by Robert Maillart (1929)*, Journal of the International Association for shell and spatial structures, 53 (2012) pp. 39-48.

[4] Bow R.H., *Economics of construction in relation to framed structures*, London, New-York, Edinburgh, E. & F.N. Spon (1873)

[5] Allen E., Zalewski W., *Form and Forces: Designing Efficient, Expressive Structures*, John Wiley & Sons, New York (2009).

[6] Autodesk, *Robot Structural Analysis*, www.autodesk.com/robot.

[7] RISA Technologies, *RISA2D*, www.risa.com/p_risa2d.html.

[8] Piker, D., *Kangaroo Physics*, www.food4rhino.com.

[9] Preisinger, C., *Karamba*, http://www.karamba3d.com.

[10] Rippmann M., Lachauer L. and Block, P., *Interactive Vault Design*, International Journal of Space Structures, Vol. 27, No. 4, (2012) pp. 219-230.

[11] Hohenwarter, M. *Geogebra*, http://www.geogebra.org.

[12] Kortenkamp, U. *Cinderella*, http://www.cinderella.de.

[13] Greenwold, S., *Active Statics*, http://ocw.mit.edu/ans7870/4/4.461/f04/module/Start.html.

[14] Van Mele T. and Block P., *eQUILIBRIUM*, block.arch.ethz.ch.

[15] Van Mele T., Lachauer L., Rippmann M. and Block P., *Geometry-based Understanding of Structures*, Journal of the International Association of Shell and Spatial Structures, Vol. 53, No. 4 (2012) pp. 285-295.

[16] Rossi F., van Beek P., Walsh T., *Handbook of Constraint Programming*, Elsevier (2006).

[17] Yates, R.C., *A Handbook of Curves and Their Properties* (1959).

[18] Ochsendorf, J., *Practice before theory: the use of the lower bound theorem in structural design from 1850-1950*, Essays: the history of the theory of structures (2005) pp.353-366.

[19] A. Muttoni, J. Schwartz, B. Thürlimann, *Design of concrete structures with stress fields* (1997).

[20] J. Heyman, *The Stone Skeleton: Structural Engineering of Masonry Architecture*, Cambridge University Press (1995).