Abstract

The work we present may be considered as the consolidation of a methodology that was already outlined in the paper presented in the second M&D congress held in San Sebastian (1998). We establish that, in architectural design, the computer is only used in the last step in order to achieve the traditional Euclidean design in a more precise and comfortable way and to improve the quality of the handmade designs.

Our proposal consists in modifying the process from the very beginning of the creative act. That is, when the design conception is born. If we want to obtain the maximum benefit of the computer possibilities, we ought to support this conception by means of a language tuned with the mentioned tool. Due to the fact that the internal language used by the computer for producing graphics is mathematical, we must incorporate, in some way, this language in the codification of the creative process.

In accordance with this setting, we propose a mathematical grammar for the design based on the construction of modulated standard mathematical functions. This grammar is developed independently from the graphical software and it is specified only when a particular computer program for the effective generation on the graphical objects is selected.

To prove and exemplify the possibilities of this methodology, we present:

1. A set of typologies or catalogues of architectural elements mathematically formulated (in the same way as a parabola or an ellipsoid is defined) and provided with several meaningful formal parameters. The structure of this catalogues allows to the users either to call a typology giving concrete values to the parameters or to produce further variations and personalized developments of the catalogue. In particular, for our communication, we have chosen the typology arch (as a generalization of a circumference) and the typology dome (as a generalization of a sphere). Moreover, if we manipulate the arches by extrusion, interpolation, etc., we can produce walls with gaps, naves, cross galleries, halls surrounded by columns, gates with archivolts, etc... We show that, with few parameters, the range of arches and domes and the combination possibilities we can produce is surprisingly ample and versatile.

2. Several complete examples developed by using these catalogues where our formulations are concreted in a specific computer language in order to be implemented in a commercial computer software package.

1. Introduction

Any creative process that aspires to go beyond a brief intuition, needs being codified in a language. This language must be represented in such a way that could be understood by oneself and, eventually, as it often happens in architectural creation, it could be transmitted to other actors of the process. Furthermore, this language must produce descriptions of the created objects that ought to be adequate to the tools used to materialize them. Thus, although the logical hierarchy assumes that is the nature of creative act (architecture, music, literature, etc.) that determines the type of codes and tools to be used, the reality is that this interaction has two directions: the tools condition the type of code but, at the same time this codification conditions the “admissible” ideas.

The traditional tools to represent the architectural design have been the rule and compasses (and their variants). In a perfect harmony with them, the language used has been (and it is) the classic Euclidean one that provides the grammatical rules to an alphabet of two letters, the right angle and the circumference’s
arch. The persistence during centuries of this efficient binomial of pairs (two tools-two letters) has engraved on the designers mind an indelible habit of thinking, of conceiving, essentially in terms of straight lines and circumference arches.

The computer science has introduced a radically different tool that it doesn’t only work synthetically, as a rule and compasses, but it works analytically: any graphical object is described by means of coordinates and mathematical functions. Until now, the job of producing this mathematical description has been left to the design program by using the information provided by the user through the respective menus. This characteristic has been exploited to produce freehand designs (Bézier, B-splines, NURBS, etc.). For the designer, working with these methods is a pure manual task. It is not a verbalized activity. To produce intelligible shapes the old Euclidean habit is always recalled. But now, the tool restrictions have disappeared, knowing its formulation, any curve or surface has the same difficulty to be produced than a straight line or a circumference. There is no excuse to carry on using an alphabet of two letters. We need to develop a new grammar and a new mentality at the moment of the creation, to overcome the old habits and take full advantage of the new tool. Having the computer a mathematical internal language it is reasonable that the suitable language to guide to user’s conceptions is a language based on mathematical terms. This language must contain the descriptions of any curve or surface by means of functions, the coordinates of the objects and a grammar allowing to functionally describing the different types of transformations and combinations of objects. The key idea is to convince the user to produce his own mathematical descriptions of the objects he is interested on and not to depend only on the instructions provided by default in the program. This way, the activity is not manual (moving points) but a verbalized one, combining functions and adjusting meaningful parameters like high, radius, curvature, etc.

Regarding to the feasibility of the designs, the structural calculus via finite elements and the current building technologies make feasible the construction of complex shapes designed by using a functional language, like the proposed in this paper that takes full benefit of the real capabilities of a computer.

2. The grammar.

In accordance with this setting, we have a proposal of a language that allows the designer to develop his creativity in a code compatible with the computer requirements.

Our initial alphabet is the set of standard functions that can be found in any graphical software and the crux point is to see this functions as the gene of a shape via a coordinate system playing the role of letters. This first alphabet is expanded by composing the preceding functions in order to obtain a wider set of shape genes. Next, every letter is converted into a word by introducing in them formally significant parameters that are adjustable by the user. This word represents a poly-parametric family of shapes available to the user. Finally, we produce a collection of functions whose role is to modify the former shapes. They are operators that actuate on the words and therefore, they are similar to verbs. They are named after the output they produce (to “periodify”, to bend, to crease, to trim, etc.) and the strength of their action is also tuned with explicit and significant parameters. The interaction between the verbs and the words, in the framework of the parametric representation of curves and surfaces produces the finished shapes. This way the finished designs are like sentences or even the stories told in this language.

We have developed a substantial number of composed letters, words and verbs. We have classified these last ones in six groups depending on the effect they produce:

- **Genetic verbs by mutation**: they work on the domain, changing the variable.
- **Surgical verbs**: they alter the image of the function.
- **Environmental verbs**: they act on the image changing the support space.
- **Dynamical verbs**: They act on the image by applying motions that are not constant in the whole support space. For example when working on a closed curve they produce a surface applying to this curve a variable motion (rotation, translation, etc).
- **Genetic verbs by crossing**: they work by combining previous shapes (interpolation, transition, juxtaposition).
- **Composed verbs**: they combine several of the previous types.

However, our aim is to propose a continuously growing open grammar where the previously built elements can be considered as starting element for further developments by adding new words and verbs like in a living language.
In the same way, we have several examples of “stories”, that is designs produce starting from this ideas, in the fields of the graphical and architectural design. Among them, we have chosen to present in this paper a rather complex one in order to show the possibilities of the proposed method. We show how to start a design from scratch and how to elaborate every functional element from the beginning showing that this work produces not only a particular design but also a package of objects to be used in future designs and can be exploited by other non expert users. After developing the package we give an example of its possibilities.

### 3. The package “arch”.

The package is structured in the modules:

- a) Module collecting the basic arches.
- b) Module of the operators that produce complex arches.
- c) Module that generates surfaces starting from arches.

Each module is open to changes and improvements and we can introduce new elements in each module. Next we give some details about them.

#### 3.1 Module defining the basic arches.

The basic requirements in order to be compatible with the other modules are:

- The arches must be in parametric form with its parameter running in the interval \([-1,1]\).
- The centre of the arch’s base must be at the origin of coordinates.
- The highest point must correspond to the value zero of the parameter.
- Outside of the interval \([-1,1]\) the function generates a “foot” along the abscises axis.

The last requirement can be obviate since in the second module there is an operator to attach “feet” to a previous arch. Any parametric curve fulfilling the mentioned conditions is a valid candidate. Among all the possible arches, we have selected

1) A catenary arch of width \(2a\) and high \(h\). (figure 1-a).
2) An arch obtained from the cosine graph in the interval \([-\pi/2,\pi/2]\) with width \(2a\) and high \(h\). (figure 1-b).
3) Arches obtained taking \(n < nt\) lobules from an epitrochoid and hypotrochoid with \(nt\) lobules. (figures 1-c and 1-d).
4) An arch of a period of a trochoid (figure 1-e).
5) A family of arches modelized from super-ellipses depending of 5 parameters. This family will be studied in detail.

![Figure 1](image)

With the aim of not being original, we choose a circumference as our basic curve. A circumference represents the arch par excellence. Centred in the origin and having radius 1, it has the well known Cartesian equation \(x^2 + y^2 = 1\). In this equation, the presence of squares prevents any discussion about signs, therefore we can write \(|x|^2 + |y|^2 = 1\). We can generalize the previous formula by changing the exponent 2 by any positive real exponent \(n\). The curves verifying the equation \(|x|^n + |y|^n = 1\) are termed super-ellipses. In figure 1 centre, we have a collection of them for different values of \(n\). For \(n \to 0\), the curve tends to a cross; for \(n=1\), it is a square with the vertexes in the axes;
for \( n=2 \), it is a circumference and for \( n \to \infty \) the curve tends to a square with its sides parallel to the axes. Taking half a super-ellipse we produce arches as seen in figure 2 right.

![Figure 2.](image)

Taking different exponents, we can generalize the previous formula and consider the curves satisfying the equation \(|x|^p + |y|^q = 1\), being \( p \) and \( q \) positive real numbers. This set of curves will be termed hiperellipses. The exponent \( q \) affects the shape of the base (figure 3 left) and the exponent \( p \) affects the shape of the cusp (figure 3 right).

![Figure 3.](image)

The next step is to give a parameterization of these curves. We have taken the following one

\[
\begin{align*}
x(u) &= \text{sign}\left(\sin\left(\frac{\pi}{2}u\right)\right) \left(\frac{\sin\left(\frac{\pi}{2}u\right)^p}{\left|\sin\left(\frac{\pi}{2}u\right)^p + \cos\left(\frac{\pi}{2}u\right)^p\right|^q}\right)^{1/p} \\
y(u) &= \text{sign}\left(\cos\left(\frac{\pi}{2}u\right)\right) \left(\frac{\cos\left(\frac{\pi}{2}u\right)^p}{\left|\sin\left(\frac{\pi}{2}u\right)^p + \cos\left(\frac{\pi}{2}u\right)^p\right|^q}\right)^{1/q} \\
u &\in [-1, 1]
\end{align*}
\]

Where \( \alpha = \max(\alpha, 2), \quad (\alpha = p, q) \). This arch passes through the points (-1,0), (0,1) and (1,0), and multiplying the \( x \) component by \( a \) and \( y \) component by \( h \), we obtain an arch of the same shape but with width \( 2a \) and high \( h \).

We can even make a further generalization. Till now, the arch has vertical studs (for \( q>1 \), the delivery of the arch at the base is always vertical). With this formulation we can’t produce horseshoe shaped arches or segmented arches that are below the circular ones. To overcome this difficulty we use a technique inspired on the NUBS curves. We situate an arch of our family on the plane \( z=1 \), we produce a shearing rescaling the vertex and projecting it again on the horizontal plane. The sequence of these operations can be seen in figure 4.
A new parameter $w$ controls this action. For $w$ positive (like in figure 4) the studs of the arch are convergent while, for negative values they are divergent. The absolute value of $w$ controls the intensity of the effect. The equations of this transformation are

\[
\begin{align*}
    x' &= \frac{x}{1 + w \cdot y} \\
    y' &= \frac{(1 + w) \cdot y}{1 + w \cdot y}
\end{align*}
\]

In figure 5, we can see an example with three values of the parameter $w$

For $p=q=2$ we obtain a circular arch but taking appropriate values of $w$, $a$ and $h$, we recover the conic curves: ellipse, parabola and hyperbola as well as any circular arch build upon a chord different from the diameter.

Summing up, we have a family of arches depending on five parameters: the width $2a$, the high $h$, $p$ and $q$ that determine the basic shape and a perspective factor $w$.

### 3.2 Module defining operators on the basic arches.

Being the arches parametric curves, their components can be the subject of functional transformations like to apply on them a verb, to compose them with other words, etc. In this module we define functions that have as its arguments arches previously defined. Its effect is to manipulate this arches to produce new composite arches. These arches could be used as input for producing other new arches. Among others we present:

- An operator that, by means of lineal transition functions adds to a previous arch two studs of variable length. The new arch is placed over the superior end of the studs. Some examples are shown in figure 6

- An operator that applies a genetic verb by crossing that uses functions of partition of the unity and that combine two previous arches. It places the second arch in the central part of the first one that has been
previously deleted. Since this action is recursive, the first arch can already be a combined one. Figure 7 shows an example.

- An operator that uses a “periodifying” verb (based on the integer part function) that allows to repeat indefinitely any arch previously defined (see figure 8). It is interesting to observe that along the periodifying process the parameter values can change and therefore, each arch of the row could be different.

![Figure 7](image1.png) ![Figure 8](image2.png)

- An operator that introduces a perspective factor $w$ to any previous arch.

- A set of operators for translating, rotating, changing the parameterization interval, changing the representation to 3D, etc., any previous arch.

### 3.3 Module to generate surfaces starting from arches.

Starting from the ample set of arches, defined using the previous modules, we can generate, by means of our grammar, surfaces that can be interesting from the architectural point of view. We have defined several operators that, when applied to arches, each one produces a different architectonical element:

- An operator that extrudes a row of arches contained in a plane along a perpendicular direction to this plane. It produces a row of naves.

- An operator that interpolates two arches placed over two parallel planes (that, eventually can coincide). This operator produces gates or walls with gaps. In the case of different planes, the two arches can be interpolated by means of the step curve obtaining a gate with archivolts.

- Operators that intersect or “cross” two extruded naves with perpendicular axes. At the crossing points we can choose among a set of different function depending on the effect we want to obtain. The functions $\max$ or $\min$ select the highest or lowest point of each nave respectively. Other operations, like t-norms can be chosen to produce other results.

We recall that, since each arch is controlled by several parameters determining its shape and measures, these parameters can change along process. That is the case of the extrusion where the parameters $p$, $q$ and $w$ could depend of the variable that generates the surface.

Next (figure 9), we show a miscellany of objects produced by means of the defined operators. The reader will easily discover which of the previous procedures is used to produce each shape.
It is worth noting that different typologies from the architectural point of view belong to the same functional typology. For example, the flat floor with pillars belongs to the same functional family that the gothic ribbed vault by changing the high of the arch and the length of the jambs.

Based on this last element that strongly suggests a gothic arch (even it is not built with circumference arches?), we have developed some full amateur designs.

4. Examples

A gothic cathedral. Starting from the last element of figure 9, by changing the parameters and by iteration, we can built gothic naves including the transept and the triforium as well as the ambulatory. By interpolating a square arch with a gothic one we obtain the closing walls. Using other resources of the grammar we have designed other elements like the gates, the flying buttresses or the pinnacles of the towers. It is specially interesting the generation of the dome that starting at its base with super-ellipse, it suffers when ascending three different transformations simultaneously. Figure 10 shows an overview of the generation of this design.
A mosque. By rotating half of an arch with respect to an axe parallel to its axe of symmetry we obtain an element like the one represented in figure 11 left. If we used a “super-rotation” the obtained object has super-elliptic sections. In figure 11 centre we can see an example with $0 < p = q < 1$. Finally, using further transformations, we obtain the object represented in figure 11 right. By means of symmetries, we assemble four copies of this last element, we build a cupola in the middle space, we repeat this shape following a rectangular grid and we obtain the design of figure 12.

Cupolas. To define the basic arch, we have started from a circumference. In the same way, starting from a sphere and following the same process we can obtain, by successive generalizations, a similar module for the design of cupolas and domes. In figure 13, we present three examples of the infinitely many possible ones.
Other designs. Apart from the typologies derived immediately from the use of the package arch that we have presented in this paper, we have developed others like the typology *step*, obtained from the function \( \arccos(\cos(x)) \), that we mentioned briefly when building archivolts. With this packages we can design a wide range of enclosures specially those devoted to receive public events like stadiums, theatres, etc. Finally, we have also experimented our grammar in the field of graphical design.

5. Conclusions.

As it has been shown along this paper, our methodology is an open one and it is still a its initial point. There are no boundaries for creating typologies and particular designs developed by means our grammar. De basic idea is to provide a technical support for describing complex shapes. If you have it, the result only depends on the creativity and sensibility of the user. Not having these capacities adds a extra limitation to the design. But, beyond the adventurers that want to experiment by themselves the possibilities of our method, there is a group of basic users that can take advantage of the packages developed by the specialists. The package will present menus to the user and the functional machinery will be transparent to them. They will only provide the values of the parameters.

Our aim is to call the attention of the real designers in order to inform them that they can conceive and create (with the process under control) with a great freedom and we have shown a way to do it. We support the idea that the grammar that we have presented resolves the problem of designing complex and singular shapes because it gives a language to clarify the ideas and to codify the precise description of the geometry of the shape. Even more, the other columns that support any architectural design apart from the design itself are the its structural complexity and its constructive complexity. Both are supported by our method through the finite elements methods used in structural calculations and by the new building technologies. With them we close the circle of the technical difficulties that have prevented the design of complex shapes.

A part from social or economical factors, there are no more excuses!.

Bibliography


