WIRIS: An Internet platform for the teaching and learning of mathematics in large educational communities

S. Xambó[1*], R. Eixarch[2] and D. Marquès[2]
1. Departament de Matemàtica Aplicada II i Facultat de Matemàtiques i Estadística. Universitat Politècnica de Catalunya
2. Maths for More, S.L., Barcelona

WIRIS is one of the main services offered by the Internet portal *edu365* of the Education Department of the Catalan Government. Its intentional users, for both individual and group activities, are the teachers and students in the secondary and primary schools of Catalonia. Access to the site is unrestricted and only a standard Web browser is needed. WIRIS supports not only advanced symbolic, geometric, and numerical computations but also the capacity to produce Web-based mathematical content. The main goal of the present paper is to explain the functionalities of that system, but we will also consider the system's design and development, its present status, how it can be easily adapted to the requirements of other communities around the world, and its perspectives for the near future.

This text is an expanded version of the invited presentation of the WIRIS system at the 27th International Symposium on Symbolic and Algebraic Computations held on July 7-10, 2002, at the Université de Lille, Lille, France (ISSAC-2002). Here, we provide many details and reflections that could not be included in the PowerPoint presentation and demo delivered in Lille on July 9, 2002 by S. Xambó. On the other hand, however, the format here only allows a description of the computations and processes that were run online at ISSAC. But of course the interested reader can use the system firsthand on the Internet. The authors would much appreciate, and acknowledge, any comments or suggestions in regard to improving the system and its uses. (NB. The original paper was composed with MS Word2000 with the American English speller.)

Acronyms

EACA	Encuentros de Álgebra Computacional y Aplicaciones
FME	Facultat de Matemàtiques i Estadística of the UPC
IEC	Institut d'Estudis Catalans
MA2	Departament de Matemática Aplicada 2 of the UPC
UPC	Universitat Politècnica de Catalunya

*Author for correspondence: Sebastià Xambó, Departament de Matemàtica Aplicada II (MA2), Universitat Politècnica de Catalunya (UPC). Jordi Girona s/n, Campus Nord, Edifici Omega. 08034 Barcelona, Catalonia (Spain). Tel. 34 934137979. Fax: 34 934017284. Email: sebastia.xambo@upc.es

1. What is WIRIS

Generically, WIRIS is an Internet platform which, on one hand, performs general mathematical computations solicited by its users and, on the other hand, supports the creation of Web-accessible interactive documents and materials.

Typical WIRIS users will be members of a community interested in getting the result of some computation, in working on some existing materials, or in producing their own documents.

In principle, there are no restrictions on the size of the community, although the installation arrangements are certainly community-dependent. Even if we could disregard, above a certain threshold, the computing and communications facilities, there would still be important aspects, such as the language or the curricula, which would have to be taken into account.

An instance of that generic system, the one with which we will mainly be concerned with here, is the Catalan version that has been running since January 2002 on a server of the Education Department of the Catalan Government. It is accessible, with no restrictions, either directly at:

http://calculadora.edu365.com

or from the main desk of the portal *edu365.* It is of interest to note that the specifications were issued by the Office for Information Technologies of the Education Department (Subdirecció General de Tecnologies de la Informació), and that later the same office took the initiative of starting a (voluntary) training program for the teachers. At the time of writing, about eight hundred secondary school teachers and about three thousand primary school teachers have attended training sessions, run mainly by other teachers trained as advance instructors.

2. First contact

The relevant part of the user interface is reproduced in Figure 1. In its caption there is a short description of the three functional areas of the interface. Here it should be remarked that for the convenience of the reader our illustrations will be based on an English version of WIRIS and not on the concrete Catalan version installed at the Education Department.

The Computation Area can contain any number of *blocks.* Initially there is just one empty block present. Blocks can be added with the icon [I on the *Edit* palette and they can be deleted (even the initial empty block) with the *Delete* key after marking the block (press the *Shift* key together with the suitable cursor keys). Each block can contain any number of expressions, each of which is signaled with a vertical bar on its left. To make room for a new expression, press the *Enter* key. An empty expression is added just before or after the active expression according to whether the cursor is at the beginning of that expression or not. Expressions can always be entered from the keyboard, but usually it is much easier, as we will see below, to use the palettes of the Command Area. Moreover, the displayed operations have a much higher typographic quality when introduced by palette. In the case of the *Operations* palette, for example, there are icons for entering fractions, powers, subindices, or roots. Note that there are also icons for grouping with parentheses, brackets, or braces.

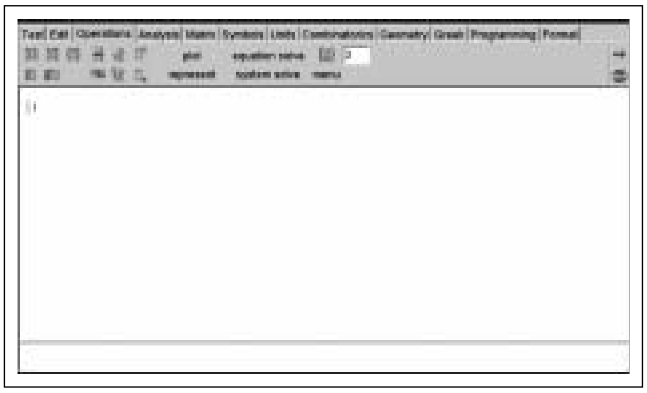
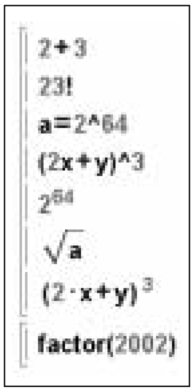

Text Edit Operations Analysis Matrix Symbols Units Combinatorics Geometry Graph Programming Format
plot
equation solve
represent
system solve
menu

Figure 1. Image of an English version of the WIRIS interface. The gray part is the *Command Area*. It displays several palettes, of which only one can be active. In this image we see that the active palette is *Operations.* The large white area below the Command Area is the *Computation Area*. Initially there is an empty block [I and it signals the place where we can enter the calculations we happen to be interested in. The narrow white area at the bottom is the *Message Area,* and this is the place where WIRIS will post its messages to the user.

The active block (the one containing the cursor) can be evaluated with the combination *Ctrl+Enter*, or by clicking on the red arrow icon. On evaluation, all the expressions in the active block, and only these expressions, are calculated, and the result of each expression is displayed to the right of it, with a red arrow in between. Notice, for comparison, that with *Enter* we get a new empty expression in the active block, while with *Cntl+Enter* we get the evaluation of all the expressions of that block.

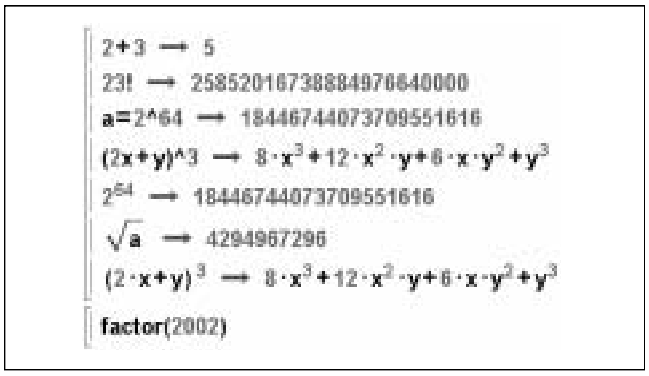
We illustrate and comment on these ideas in Figures 2 and 3, where we display only the relevant part of the Computation Area. On the left-hand side of Figure 2 we can see a Computation Area with two blocks, the first with seven expressions and the second with only one expression. On the left-hand side of Figure 3 we can see the result of evaluating the first block. In Figure 2, the third and fifth expressions have the same value, but they were entered differently. For the third, only the keyboard was used, while in the fifth we resorted to the exponent icon of the *Operations* palette. A similar situation occurs for the forth and seventh expressions. In the sixth expression the useful item is the square root icon. It is important to remark that integers can have as many digits as wanted (at least for all practical computational purposes).



$2+3$
23!
a=2^64
(2x+y)^3
2^{64}
$\sqrt{a}$
$(2\cdot x+y)^3$

factor(2002)

Figure 2. In the display on the left there are two blocks. The first contains seven expressions and the second only one. The first four expressions of the first block have been entered with the keyboard only, while in the other three the exponent, square root, and parenthesis icons have been used. Note also the 2x in the forth expression (it will be interpreted as 2 times x) and the 2·x in the seventh, where the product operation has been entered explicitly. When we make the first block active and evaluate it, we will get the values of its seven expressions, but not of the expression in the second block. The result is illustrated in figure 3.



$2+3 \longrightarrow 5$
$23! \longrightarrow 25852016738884976640000$
a=2^64 $\longrightarrow 18446744073709551616$
(2x+y)^3 $\longrightarrow 8\cdot x^3+12\cdot x^2\cdot y+6\cdot x\cdot y^2+y^3$
$2^{64} \longrightarrow 18446744073709551616$
$\sqrt{a} \longrightarrow 4294967296$
$(2\cdot x+y)^3 \longrightarrow 8\cdot x^3+12\cdot x^2\cdot y+6\cdot x\cdot y^2+y^3$

factor(2002)

Figure 3. Here we have the results of evaluating the first block of the previous display. Note that the value of the third and fifth expressions are the same, as are the values of the fourth and seventh. On the other hand, the second block has remained unevaluated. The integers can have, for all practical purposes, as many digits as wanted.

3. Basic philosophy

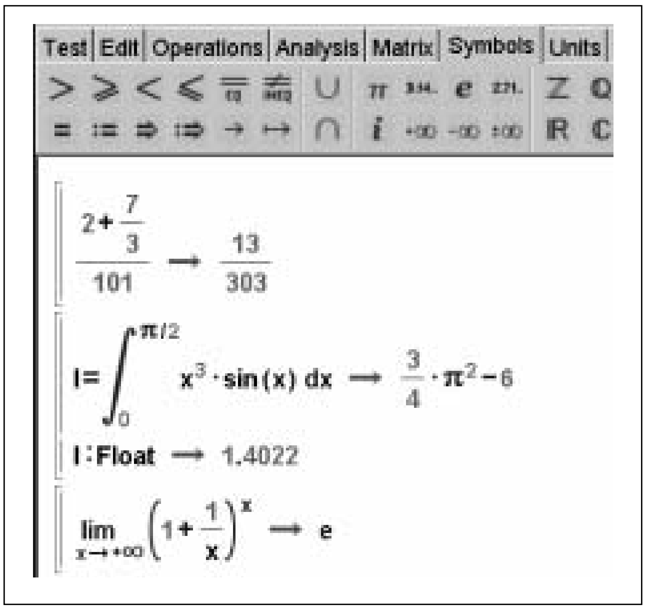
Briefly expressed, the motto is that in the programming of a system for doing mathematics by computer «mathematical language should be the ruler» (at least as much as possible).

This is not easy, because mathematical language is quite complex, but the point behind the thought is that the syntax and semantics of mathematical expressions, including those of a logical nature, have evolved through the centuries into a very expressive, concise, and reliable language. Hence, substantial advantages are gained by making as much of this language as possible intelligible to a computer program.

This position is contrasts with the design of most programming languages. In fact, if it is true that every programming language takes on some features of that ideal mathematical language, especially the logical aspects, it is also clear that most of those features are, generally speaking, disregarded. Thus programming languages usually provide a way of representing integers, like 314, but rarely care about having a representation, just as another object, of the set of all integers, together with the integer's Euclidean ring structure. And much less attention is paid to using the standard symbol *Z* to denote that structure.

At the other end of the spectrum, programming languages often include idiosyncratic constructs that are extraneous to mathematics proper, even languages that at a first glance would seem to follow our motto. For example, while the names of functions in mathematics are conveniently arbitrary for all practical purposes, like f, G, or mu, there are systems that, quite arbitrarily, only allow using names that begin with a capital letter, or perhaps just the other way around. The value of a function, say mu, on a number, n, is customarily represented mu(n) in a mathematical context, but there are systems that instead of (n) expect [n], and so on.

In the design of WIRIS we have followed our motto as faithfully as possible, and will endeavor to keep doing so in the future. In particular, we use standard notations and names whenever possible, like *Z* for the ring of integers and *Q*, *R*, and *C* for the fields of rational, real and complex numbers, respectively. The palette symbol *i* (which is not to be confused with the identifier i) stands for the imaginary unit, π for the exact real number pi, and **8** for infinity. WIRIS understands that integers are also rational numbers, as in mathematics. Similarly, rational numbers are seen, when appropriate, as real numbers



$$\frac{2+\frac{7}{3}}{101} \longrightarrow \frac{13}{303}$$

$$I=\int_0^{\pi/2} x^3 \cdot \sin(x)\,dx \longrightarrow \frac{3}{4}\cdot\pi^2-6$$

$$I{:}\mathrm{Float} \longrightarrow 1.4022$$

$$\lim_{x\to+\infty}\left(1+\frac{1}{x}\right)^x \longrightarrow e$$

Figure 4. In the display the active palette is *Symbols* and we have three blocs. The value of the expression in the first block is a rational number (a fraction), but note, for example, that the numerator is the sum of an integer (2) and a rational number (7/3), so that the natural transformation of 2 into 2/1 is performed automatically. This behavior is actually a general feature of WIRIS which is implemented in most of its operators (we will see more examples below). In the second block we have a definite integral, which has been entered with the *Analysis* and *Symbols* palettes). Note that the integral gives an exact real number I, and that the approximate decimal value can be found with the expression I:Float. The last bloc is a well-known limit, entered using the *Operations* and *Symbols* palettes.

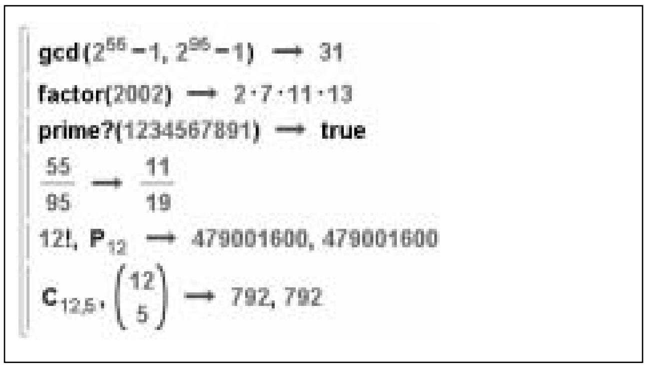
and real numbers as complex numbers. This illustrates the fact that the system «knows» quite complex structures and their interrelations. Some of these features are illustrated in Figure 4. In the following sections we will see many other aspects of this mathematical-like behavior. In general, the natural conversion of an object x, say of type A, into an object of type B is obtained, if the conversion A –> B is defined, with the expression x:B. If n is an integer, for example, n:*C* is n seen as a complex number. Similarly, if r is a rational number, a radical or a real number, r:Float yields the decimal approximation of r in the current precision (which can be changed, as we will see in some of the displays, as for example in Figure 12, with the command *precision*). The same result would be obtained with the expression 1.0 · r (as 1.0 is a Float, r is converted, before evaluating the product, into a Float).

4. Mathematical capabilities

In this section, we succinctly describe the most important objects and mathematical capabilities that WIRIS can handle. Below, we will illustrate some of these capabilities with representative displays.

Integers, rational numbers, combinatorial numbers

Integers, the usual operations with integers and divisibility functions (greatest common divisor (gcd), least common multiple (lcm), primality testing and factorization). Rational numbers (usual operations, automatic reduction to irreducible form). In the combinatorial context we can compute the number of permutations (factorial), variations and combinations (binomial numbers). See Figure 5.



$$\mathbf{gcd}(2^{55}-1, 2^{95}-1) \longrightarrow 31$$

$$\mathbf{factor}(2002) \longrightarrow 2\cdot7\cdot11\cdot13$$

$$\mathbf{prime?}(1234567891) \longrightarrow \mathbf{true}$$

$$\frac{55}{95} \longrightarrow \frac{11}{19}$$

$$12!,\ \mathbf{P}_{12} \longrightarrow 479001600,\ 479001600$$

$$\mathbf{C}_{12,5},\ \binom{12}{5} \longrightarrow 792,\ 792$$

Figure 5. The function gcd yields the greatest common divisor of two integers and factor yields the prime decomposition of an integer. The third expression tells us that 1234567891 is a prime number and the fourth illustrates the automatic reduction of fractions to irreducible form. The last two lines show the calculation of two combinatorial expressions. Except 12!, those expressions have been composed with the combinatorics palette.

Sequences, lists, vectors, relations, divisors, tables and ranges

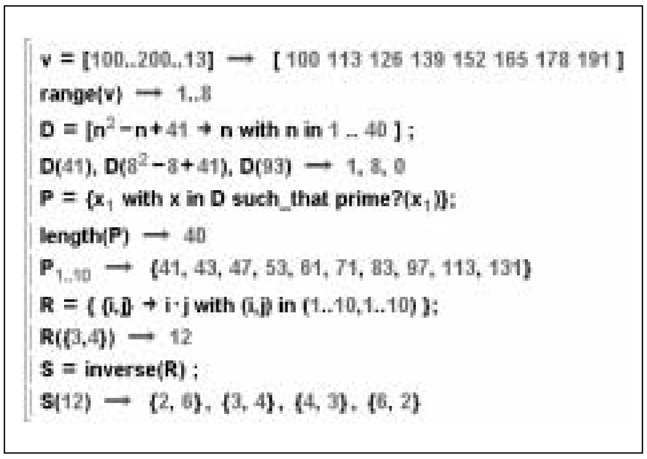
These are various ways that can be used to group objects (see Figure 6). A sequence is an ordered succession of objects separated by commas. Lists and vectors are sequences enclosed in braces and brackets, respectively. For example, {1, 2, x} is a list and [1, 2, x] is a vector. Relations and divisors are lists of objects of the form a (ρ)

b, where a and b can be arbitrary objects, enclosed in braces and brackets, respectively. For example, {a (ρ) 1, b (ρ) 2, c (ρ) x} is a relation and [a (ρ) 1, b (ρ) 2, c (ρ) x] is a divisor. Tables are lists of objects of the form x = a, where x is an identifier and a an arbitrary object. For example, {x = 3, y = 2, z = 1} is a table. Ranges are expressions of the form a..b or a..b..c. The expression a..b..c can be understood as a representation of the sequence of values of the form a+jc that lie in the interval [a,b] and with j a non-negative integer, whereas a..b represents a sequence of values in the interval [a,b] that are computed according to diverse rules depending on the context, that is, on the function using that range. For example, [a..b] is equivalent to [a..b..1], but in plot(f,a..b), where f is a function defined in the interval [a,b], a..b is interpreted as a sequence of values that in general depend on f.

All the groupings considered in the last paragraph can be constructed and assembled together in quite natural ways and often with a familiar mathematical-like syntax. It is to be remarked that finite sets are represented as lists of distinct elements and that the basic operations with sets (union, intersection, difference) are supported (see the palette *Symbols* in Figure 4). Finally, combinatorial expressions are overloaded so that they may yield sets when a suitable argument is a set (or list) instead of a number. For example, while the value of P_3 is 3! = 6, $P_{\{1,2,x\}}$ yields the set of the 6 permutations of the set {1, 2, x}.

Real and complex numbers

Radicals (with automatic normalization). Decimal numbers (or Floats). Real numbers (including exact constants such as π and *e*). Complex numbers. Transcendent functions of a real variable (trigonometric functions, exponentials and logarithms). See Figures 7.



$v = [100..200..13] \longrightarrow [\,100\ 113\ 126\ 139\ 152\ 165\ 178\ 191\,]$

$\text{range}(v) \longrightarrow 1..8$

$D = [n^2-n+41 \rightarrow n \text{ with } n \text{ in } 1..40\,];$

$D(41),\ D(8^2-8+41),\ D(93) \longrightarrow 1,\ 8,\ 0$

$P = \{x_1 \text{ with } x \text{ in } D \text{ such\_that prime?}(x_1)\};$

$\text{length}(P) \longrightarrow 40$

$P_{1..10} \longrightarrow \{41, 43, 47, 53, 61, 71, 83, 97, 113, 131\}$

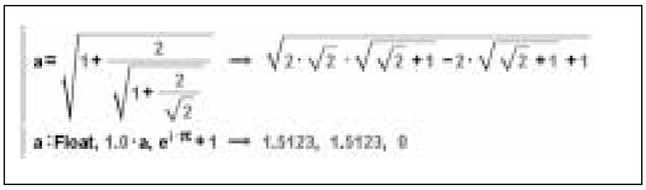
$R = \{\,(i,j) \rightarrow i\cdot j \text{ with } (i,j) \text{ in } (1..10, 1..10)\,\};$

$R(\{3,4\}) \longrightarrow 12$

$S = \text{inverse}(R)\,;$

$S(12) \longrightarrow \{2, 6\}, \{3, 4\}, \{4, 3\}, \{6, 2\}$

Figure 6. The first expression in the display gives the vector v associated to a range, and the second gives the range of v (has 8 components). The next five expressions play on the fact, observed by Euler, that the polynomial n^2-n+41 has 40 consecutive values that are prime numbers. D is a divisor that associates n to n^2-n+41 for n in 1..40 and P is the list of the n^2-n+41 that are prime (all are, as corroborated by length (P)=40). In the last four expressions, we construct the multiplication table as a relation, and then find the ways 12 is obtained.



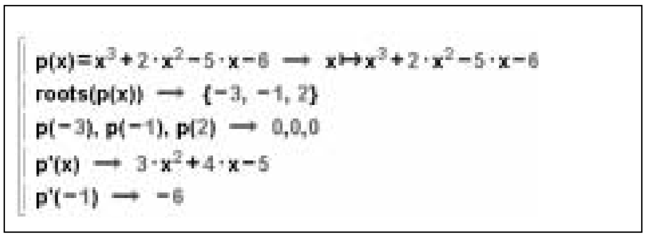
$a = \sqrt{1+\dfrac{2}{\sqrt{1+\dfrac{2}{\sqrt{2}}}}} \longrightarrow \sqrt{2\cdot\sqrt{2}\cdot\sqrt{\sqrt{2}+1}-2\cdot\sqrt{\sqrt{2}+1}+1}$

$a : \text{Float},\ 1.0\cdot a,\ e^{i\cdot\pi}+1 \longrightarrow 1.5123,\ 1.5123,\ 0$

Figure 7. Automatic simplific-ation of a radical and its dec-imal approximation (done in two ways). Also a well-known identity of Euler.

Polynomial algebra

Polynomials with numeric coefficients (integer, rational, decimal or complex numbers). Polynomials with symbolic coefficients (parameters). Divisibility functions (gcd, lcm, primality and factorization). Roots of polynomials (integer, rational, radical, decimal or complex roots). Solution of polynomial equations (that may contain parameters). Rational functions (operations and decomposition into simple fractions). See Figure 8.



$p(x) = x^3+2\cdot x^2-5\cdot x-6 \longrightarrow x \mapsto x^3+2\cdot x^2-5\cdot x-6$

$\text{roots}(p(x)) \longrightarrow \{-3, -1, 2\}$

$p(-3),\ p(-1),\ p(2) \longrightarrow 0,0,0$

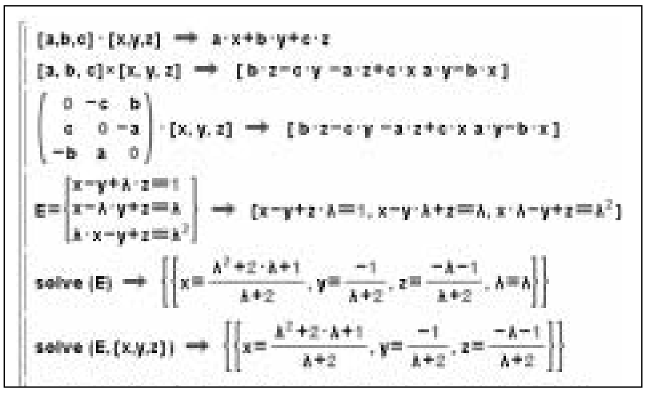
$p'(x) \longrightarrow 3\cdot x^2+4\cdot x-5$

$p'(-1) \longrightarrow -6$

Figure 8. The first expression defines a polynomial of degree 3 in the variable x. The second expression finds its roots, which happen to be –3, –1 and 2 (integers in this case). The third expression checks that they are indeed roots. The forth computes the derivative of p(x), p'(x), and its value at –1.

Linear algebra

Vector and matrix algebra (vector space structures, dot and cross products, matrix operations). The entries of vectors and matrices can be symbolic expressions. Rank of a matrix. Determinant and trace of a square matrix. Solution of systems of linear equations, possibly with parameters (the equations can be in the standard mathematical form or we



$[a,b,c]\cdot[x,y,z] \longrightarrow a\cdot x+b\cdot y+c\cdot z$

$[a,b,c]\times[x,y,z] \longrightarrow [\,b\cdot z-c\cdot y\ \ -a\cdot z+c\cdot x\ \ a\cdot y-b\cdot x\,]$

$\begin{pmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{pmatrix}\cdot[x,y,z] \longrightarrow [\,b\cdot z-c\cdot y\ \ -a\cdot z+c\cdot x\ \ a\cdot y-b\cdot x\,]$

$E = \left\{\begin{matrix} x-y+\lambda\cdot z = 1 \\ x-\lambda\cdot y+z = \lambda \\ \lambda\cdot x-y+z = \lambda^2 \end{matrix}\right] \longrightarrow [x-y+z\cdot\lambda = 1,\ x-y\cdot\lambda+z = \lambda,\ x\cdot\lambda-y+z = \lambda^2]$

$\text{solve}(E) \longrightarrow \left\{\left[x = \dfrac{\lambda^2+2\cdot\lambda+1}{\lambda+2},\ y = \dfrac{-1}{\lambda+2},\ z = \dfrac{-\lambda-1}{\lambda+2},\ \lambda = \lambda\right]\right\}$

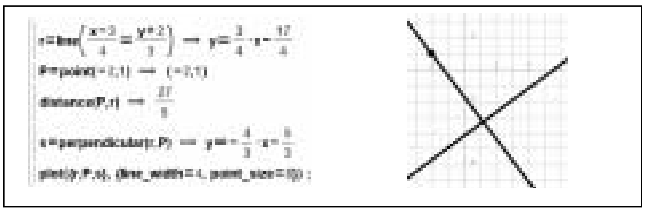
$\text{solve}(E,\{x,y,z\}) \longrightarrow \left\{\left[x = \dfrac{\lambda^2+2\cdot\lambda+1}{\lambda+2},\ y = \dfrac{-1}{\lambda+2},\ z = \dfrac{-\lambda-1}{\lambda+2}\right]\right\}$

Figure 9. Here we have the dot and cross product of two symbolic vectors and the interpretation of the cross product as the action of a skew-symmetric matrix. Then we have defined and solved a system E of three linear equations in three unknowns and whose coefficients are functions of a parameter λ (note the two calls of *solve*) . Finally, we have specialized E for λ=-2 and found that the resulting system has no solutions (it is incompatible).

can use matrix notations). See Figure 9. In general, the function *solve* returns a list of tables, where each table represents one solution. In the case of the equations E in Figure 9, there is a unique solution and so the list contains just one table.

Geometry and graphics

Primitives for the creation of geometrical 2D objects (points, vectors, lines, segments, triangles, polygons, circumferences, arcs, conics and curves, among others). Operations with geometric figures (distance, angle, intersection, functional representation of geometrical transformations). Automatic conversion of equations into geometric objects. Conversions between the different sorts of equations of a line (explicit, implicit, point-slope). Graphical representation for all geometrical figures. Flexible and powerful facilities for managing the plotter attributes. Graphics can be exported in the pdf and ps formats. See figures 10 and 11.



$r=\text{line}\left(\frac{x-3}{4}=\frac{y+2}{3}\right) \longrightarrow y=\frac{3}{4}\cdot x-\frac{17}{4}$

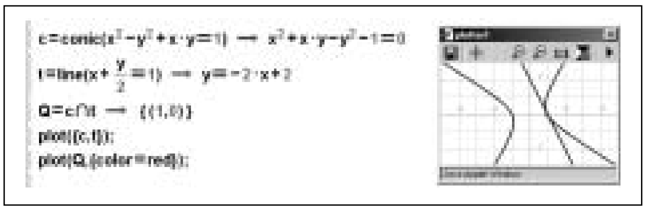
$P=\text{point}(-2,1) \longrightarrow (-2,1)$

$\text{distance}(P,r) \longrightarrow \frac{27}{5}$

$s=\text{perpendicular}(r,P) \longrightarrow y=-\frac{4}{3}\cdot x-\frac{5}{3}$

plot({r,P,s}, {line_width=4, point_size=8}) ;

Figure 10. The first expression creates a line r from its implicit equation. The second creates the point (-2,1) and the third computes the distance of P to r. The perpendicular to r through P is called s (note that its value is returned as an explicit equation), and finally we draw r and P in rather thick pens.



$c=\text{conic}(x^2-y^2+x\cdot y=1) \longrightarrow x^2+x\cdot y-y^2-1=0$

$t=\text{line}(x+\frac{y}{2}=1) \longrightarrow y=-2\cdot x+2$

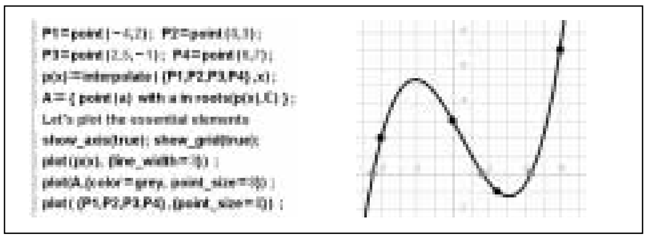
$Q=c\cap t \longrightarrow \{(1,0)\}$

plot({c,t});

plot(Q,{color=red});

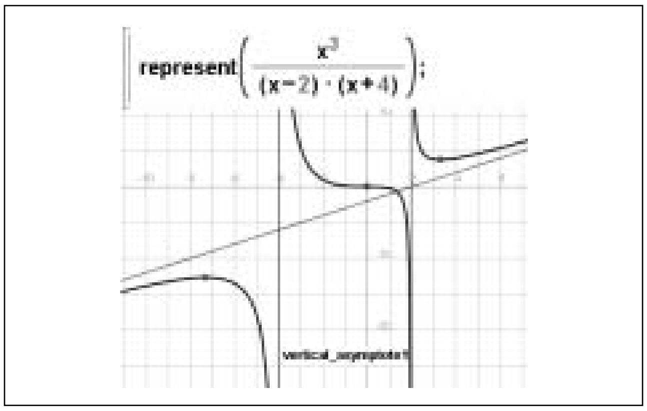
Figure 11. The first expression creates a conic c from its implicit equation. The second creates a line from its implicit equation. The intersection of c and t is called Q, and it turns out to consist of one point, namely (1,0). Finally, we plot Q in red and c and t with the default attributes.

Graphics support interactivity in a very simple way, as explained in Figure 12.



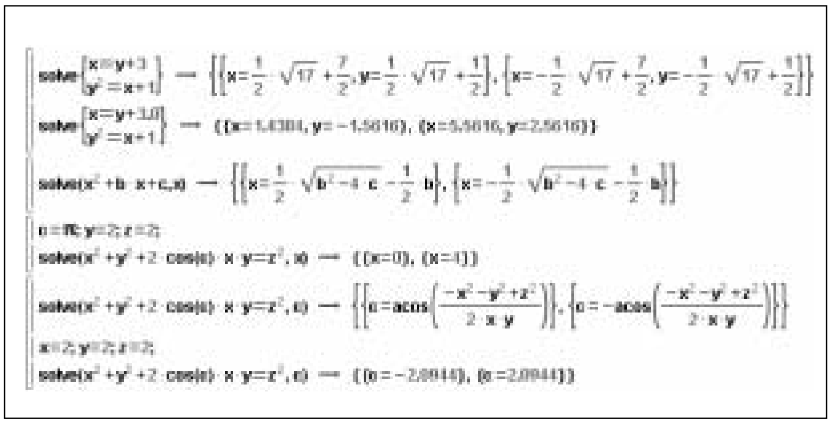
```
P1=point(-4,2); P2=point(0,3);
P3=point(2.5,-1); P4=point(5,7);
p(x):=interpolate({P1,P2,P3,P4},x);
A:={point(a) with a in roots(p(x),ℂ)};
Let's plot the essential elements
show_axis(true); show_grid(true);
plot(p(x), {line_width=3}) ;
plot(A,{color=grey, point_size=9}) ;
plot({P1,P2,P3,P4},{point_size=6}) ;
```

Figure 12. Here we first define four points P1, ..., P4. Then we let p(x) denote the polynomial that interpolates these points (it is the polynomial of minimum degree whose graph passes through the points), and A be the set of points corresponding to the (complex) roots of p(x). Finally, we plot p(x), then the roots in fat gray points and then the points P1, ..., P4 in black. The interactivity of the WIRIS graphics means, in this case, that if we drag either of the points P1, ..., P4 with the mouse, then both p(x) and A are recomputed and the plot refreshed, so that we can see how the graph and its roots evolve when the data P1, ..., P4 are moved. This is particularly striking and enlightening when two roots become complex conjugate, as in that case these two roots wander away from the x-axis. The key point in this interactive mechanism is that p(x) and A are defined with the := operator (and not just =), and this is what makes the reevaluating of p(x) and A happen any time the initial date are modified.



$\text{represent}\left(\frac{x^3}{(x-2)\cdot(x+4)}\right);$

Figure 13. Here we can see some of the features of the represent command. It finds the more relevant elements of the graph of a function, such as maxima, minima, inflection points and asymptotes. It also selects the axes and the corresponding units, and finally plots all these elements. When we explore with the cursor the distinct graphical elements, we get information on the nature of those elements (in the display, the *vertical_asymptote1* appeared when the blue line x=-2 was pointed to with the mouse arrow.



$\text{solve}\begin{bmatrix}x=y+3\\y^2=x+1\end{bmatrix} \longrightarrow \left\{\left[x=\frac{1}{2}\cdot\sqrt{17}+\frac{7}{2}, y=\frac{1}{2}\cdot\sqrt{17}+\frac{1}{2}\right],\left[x=-\frac{1}{2}\cdot\sqrt{17}+\frac{7}{2}, y=-\frac{1}{2}\cdot\sqrt{17}+\frac{1}{2}\right]\right\}$

$\text{solve}\begin{bmatrix}x=y+3.0\\y^2=x+1\end{bmatrix} \longrightarrow \{\{x=1.4384, y=-1.5616\},\{x=5.5616, y=2.5616\}\}$

$\text{solve}(x^2+b\cdot x+c,x) \longrightarrow \left\{\left\{x=\frac{1}{2}\cdot\sqrt{b^2-4\cdot c}-\frac{1}{2}\cdot b\right\},\left\{x=-\frac{1}{2}\cdot\sqrt{b^2-4\cdot c}-\frac{1}{2}\cdot b\right\}\right\}$

$a=\pi; y=2; z=2;$

$\text{solve}(x^2+y^2+2\cdot\cos(a)\cdot x\cdot y=z^2,x) \longrightarrow \{\{x=0\},\{x=4\}\}$

$\text{solve}(x^2+y^2+2\cdot\cos(a)\cdot x\cdot y=z^2,a) \longrightarrow \left\{\left\{a=\text{acos}\left(\frac{-x^2-y^2+z^2}{2\cdot x\cdot y}\right)\right\},\left\{a=-\text{acos}\left(\frac{-x^2-y^2+z^2}{2\cdot x\cdot y}\right)\right\}\right\}$

$x=2; y=2; z=2;$

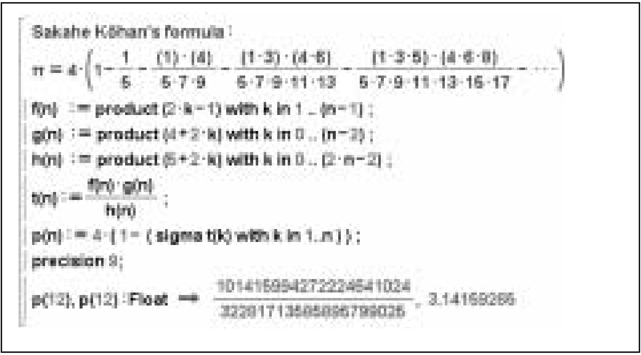
$\text{solve}(x^2+y^2+2\cdot\cos(a)\cdot x\cdot y=z^2,a) \longrightarrow \{\{a=-2.0944\},\{a=2.0944\}\}$

Figure 14. Examples of solving non-linear equations. In the first block we ask for the solution of two polynomial equations that only differ in one additive coefficient (the integer 3 in the first case and the float 3.0 in the second). In the first case we get the exact solutions, expressed as radicals, and in the second we get decimal approximations of the solutions. In the second block we ask for the solution of a quadratic equation, but with indeterminate coefficients, and the result is equivalent to the quadratic formula for the solution of those equations. The third block is actually a quadratic equation, because cos(π)= –1, and we get the exact solution. In the last block we solve two similar non-polynomial equations, in the first case with x,y,z as parameters and in the last as concrete values. Since the evaluation of acos cannot be done exactly, it is approximated with decimal numbers.

Analysis
Limits. Symbolic derivatives. Taylor polynomials. Symbolic integration, including integrands dependent on parameters. Definite integrals. Smart facilities for the representation of the graph of functions and their main elements (maxima and minima, inflections, symmetries, asymptotes). See Figure 13.

Other aspects
Progressions (arithmetic and geometric). *Series. Infinity* (the infinity symbol and its roles and computation rules). *Statistics* (relative and absolute frequencies; mean, standard deviation, median, quartiles; linear regression; histogram, box diagrams, bar graphics; other representations). *Units* (the SI system of units; other systems of units and user defined units; conversions). *Solving equations* (solving systems of equations, linear and non-linear; numerical solution of systems of equations; solving inequality systems in one variable). See Figure 14.

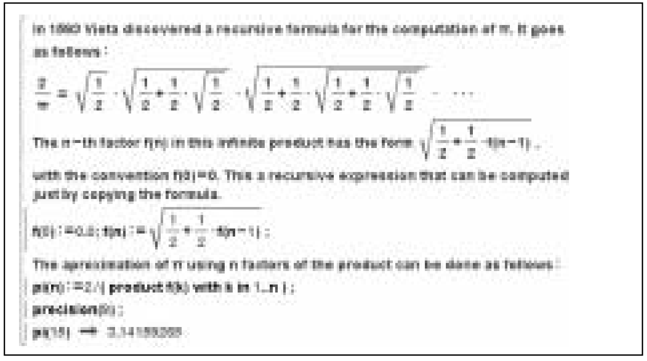
Programming
High-level, mathematically friendly, programming language. Includes some of the most basic advantages of various programming paradigms (imperative, structured, functional). Flexible modularity facilities, user types, overloading of all functions (including the kernel functions), optional checking of function parameters beyond types. Figures 15 and 16 illustrate some of these features.


Sakahe Köhan's formula :

$$\pi = 4\cdot\left(1-\frac{1}{5}-\frac{(1)\cdot(4)}{5\cdot7\cdot9}-\frac{(1\cdot3)\cdot(4\cdot6)}{5\cdot7\cdot9\cdot11\cdot13}-\frac{(1\cdot3\cdot5)\cdot(4\cdot6\cdot8)}{5\cdot7\cdot9\cdot11\cdot13\cdot15\cdot17}-\cdots\right)$$

f(n) := product (2·k−1) with k in 1 .. (n−1) ;
g(n) := product (4+2·k) with k in 0 .. (n−2) ;
h(n) := product (5+2·k) with k in 0 .. (2·n−2) ;
$t(n) := \frac{f(n)\cdot g(n)}{h(n)}$;
p(n) := 4·(1 − (sigma t(k) with k in 1..n)) ;
precision 9;
p(12), p(12) :Float $\longrightarrow \frac{10141695427222464 1024}{32281713585896799025}$, 3.14159265

Figure 15. The samurai Sakahe Köhan (1729-1824) discovered a nice formula for the computation of π. It is shown in the display above, together with five auxiliary functions to compute it. The functions f, g and h yield the two factors in the numerator and the denominator of the n-th negative term in the parenthesis of the formula, and t yields the n-th term itself. Finally, p(n) computes the formula up to the n-th term. For n=12 it gives eight exact decimal digits.


In 1593 Vieta discovered a recursive formula for the computation of π. It goes as follows :

$$\frac{2}{\pi} = \sqrt{\frac{1}{2}}\cdot\sqrt{\frac{1}{2}+\frac{1}{2}\cdot\sqrt{\frac{1}{2}}}\cdot\sqrt{\frac{1}{2}+\frac{1}{2}\cdot\sqrt{\frac{1}{2}+\frac{1}{2}\cdot\sqrt{\frac{1}{2}}}}\cdot\cdots$$

The n−th factor f(n) in this infinite product has the form $\sqrt{\frac{1}{2}+\frac{1}{2}\cdot f(n-1)}$,
with the convention f(0)=0. This a recursive expression that can be computed just by copying the formula.
f(0) :=0.0; $f(n) := \sqrt{\frac{1}{2}+\frac{1}{2}\cdot f(n-1)}$;
The aproximation of π using n factors of the product can be done as follows :
pi(n) :=2/(product f(k) with k in 1..n) ;
precision(9) ;
pi(15) $\longrightarrow$ 3.14159265

Figure 16. The display above illustrates that recursive functions are supported. Note also that since the value of f(0) is 0.0 (a Float), the value of f(n) will automatically be a float for all n. For n=15, we get eight exact decimal digits of π. If instead of f(0):=0.0 we had written f(0):=0, then f(n), and hence pi(n), would be expressed in nested radicals, as in Vieta's formula.

5. Content generation

Content can be generated with the WIRIS editor, as was done with all the examples displayed in this paper. In Figure 13, for instance, there is a block of text, then an expression, then another line of text, and, finally, three more expressions. All this was composed with the editor. Notice that the formulas, thanks to the palette's icons, were as easy to write as the pure text.

Once a piece of work is finished, it can be saved, both locally or on user-accessible servers, in html format. Thus, these files can be recovered with any standard Web browser and included in other html documents. It is important to note that these embedded segments preserve all the WIRIS functionality, as for example interactivity.

6. Advantages for users and for the community

As stated above, only a standard Web browser with Java is required on the part of the end user. In other words, users need no other software.

The system is simple and powerful. Mainly because of the smart and intuitive facilities of its interface, it is easy to use and easy to learn. Those facilities provide a remarkable integration of the computational engine with standard mathematical language, and we believe that this has many interesting and positive consequences for the teaching and learning of mathematics. For example, it favors naturally the incremental construction of structured content repositories, both for individuals and for communities. It also fosters experimenting with mathematical ideas, including of course geometry and graphics, an aspect unfortunately much neglected by the so-called *New Maths*.

For the educational community, it is easy to adapt to the curricula and the relevant legal regulations. The interface and the computational engine can also be adjusted to the specifications of the community. In any case, the architecture of WIRIS promotes an excellent adjustment to the computer and communications facilities available. In particular, its performance improves automatically when the servers or the telecommunications nets are upgraded. In addition, we would like to stress that the cost per community user is very small, because the community servers need not be special in any way (500 MHz and 256 RAM should be enough in most situations) and, as already stated, individual users only need to have access to a Web browser.

7. Research and development

The research required to create a «Web platform enabling a whole teaching-learning community to do general mathematical computations and to produce Web-based content repositories» has been long and multidisciplinary. The main aspects are the computational engine, which is responsible for processing the mathematical expressions and evaluating them, and the interface that manages the communications between the user and the computational engine. Let us consider each in turn.

Computational engine

This has been developed over the course of the last eight years, first as a program (Ω_0) to handle the mathematically intensive computations in the theory of algebraic error-correcting codes and subsequently as a general system for computer mathematical manipulations (Ω and *OmegaMath*).

The research, design, and production of Ω_0 was led by S. Xambó and was carried out with the collaboration of J. Mola, D. Marquès and A. Buil, at that time undergraduate students at the FME of the UPC (see [1]). The good performance of Ω_0 sparked the idea of building Ω, a general symbolic mathematical manipulator.

The research and development of Ω was again led by S. Xambó and was carried out with the collaboration of D. Arso, M. Castells, R. Eixarch, P. Garriga, J. Mola, D. Marquès and X. Vindel. The system was designed to be strongly typed, with user definable types, and to include the essential characteristics of functional languages, including recursivity, and also the familiar and useful constructions of imperative and structured languages. All this involves delicate and complex issues (the interested reader might like to consult the references [2], for example). As expressed earlier in this paper, another requirement was that the system should, as much as possible, be able to handle the usual formulae, both in syntax and semantics, of basic mathematical language.

The Ω phase was labeled Project OMEGA, and one of its main aspects was the development of a number of *Technological Projects* at the FME, each of which was a contribution to some of the Ω modules (see [3]). In relation to Ω, the project of D. Marquès turned out to be a cornerstone. It was a program written in Haskell that was endowed with most of the desired features and which allowed experimentation with different syntaxes and semantics, thus permitting us to be able to decide which one was the most appropriate (see [4]). The result, completed at the beginning of 1998, was a program that was called *OmegaMath*, and whose development was pursued in the FME until July 1999. Since then the development has been carried out by the company *Maths for More*, of which we will say a bit more in a moment, with a key role played by Daniel Marquès as a technological director. The computational engine behind WIRIS is a mild specialization of *OmegaMath* with regard to the purposes it has to serve.

Interface

The end users (teachers, students, parents, etc.) access WIRIS with a standard Web browser, but the real job is done by a Java applet**.** This applet is the real power-horse behind the WIRIS interface and is responsible of three main tasks. One of these tasks is rendering the documents onscreen with high quality mathematical typography. A second task, closely related to the first, is that it includes an editor that enables the user to produce new materials, or to modify old ones. The final task, which conceptually is the most important, is the translation of the expressions solicited for evaluation into a form which will be understood by the computational engine and, conversely, to present the answers of the computational engine to the user in a suitable form. The WIRIS interface Java applet has been developed entirely at *Maths for More***,** in a project commissioned to Pau Gargallo.

Maths for More was created by the authors**,** starting with *OmegaMath*, in the context of the Innova program of the UPC. The goal of Innova is to favor the launching of companies based on ideas and developments of university members, and the *Maths for More* mission is to design and produce useful computer programs for the teaching and learning of mathematics, and also for carrying out mathematical research (see [5]). The WIRIS interface is being improved continually. Because of the nature of the system, any improvement benefits all users immediately, as only one program in one server needs to be upgraded.

8. Perspectives

We believe that WIRIS provides new ways and opportunities for teachers to experiment with and engage in research on the important problems connected with the teaching and learning of mathematics. There are many ideas about how to improve the quality of teaching and learning, including the overall «educational throughput», which appear in a new light when systems like WIRIS become available. Experimenting, both individually and collectively, becomes easy and attractive, even playful to a great extent, and it is well known that true knowledge is only acquired by doing, by making each subject experience what has to be learned.

On the other hand, WIRIS enables users to focus, at each learning stage, on the essential issues of that stage, and not on issues that properly pertain to other stages. Last but not least, not only avoids having to perform tedious and unenlightening computations, but allows carrying out larger and more ambitious computations that could otherwise not even be dreamt of.

WIRIS is a tool to help the thinking of mathematics. Just as trains or planes have changed our views of the world, or telescopes of the universe, mathematical systems such as WIRIS will favor a more conceptual teaching and learning of mathematics because they make possible more detailed explorations, in much less time, of wider territories.

For information on research done with WIRIS, we refer the reader to [6] and [7].

Acknowledgments

We wish to express our gratitude to the FME and the MA2 for encouraging the ideas and developments related to Project

OMEGA, and to the UPC for the Innova Program, which guided us through valuable initiation rituals related to entrepreneurship. It is also a pleasure to thank the Department of Education of the Catalan Government for their confidence in supporting new technologies in education. Finally, we are especially grateful for the long cooperation with Ferran Ruiz and Jordi Castells on the design of WIRIS**,** and for the training program and diffusion arrangements set up by Antoni Gomà.

References

[1] S. Xambó y J. Mola: (I) Omega: *Engineering a System for Effective Computations related to Block Error-correcting Codes*. (II) : *Problem Solving with the Interpreter* Ω_0. Proc. III International Congress of Project Engineering, eds. A. Creus, J. Forès, F. Tadeusz and J. Aragonès (AEIPRO, 1996), 1518-1525 y 1697-1704, respectively. At the same Congress, a poster that summarized the characteristics of the system was presented (signed by S. Xambó, J. Mola, D. Marquès and À. Buil).

[2] Carlo Ghezzi y Medhi Jazayeri: *Programing Language Concepts* (3rd Ed.), Wiley, 1998. Kim B. Bruce: *Foundations of Object Oriented Languages* (Types and Semantics). The MIT Press, 2002.

[3] Here is the list of Technological Projects OMEGA, and two related projects, directed by S. Xambó:
- X. Vindel, *Computacional theory of groups* (May 1998).
- D. Marquès, *MEGA: A symbolic manipulator prototype* (September 1998).
- D. Arso, *Finite fields and polynomials* (September1998).
- R. Eixarch, *The module of integers and rational numbers* (July 1999).
- M. Castells, *The geometry and graphics module* (July 2000).
- P. Garriga, *The linear algebra module* (July 2000).

Related projects:
- L. Garcia, B. Llacay, *The compact disc codes* (July 2000).
- G. Huguet, *The integration module* (scheduled to appear in March 2003).

[4] For the functional aspect of these works, see S. Xambó and D. Marquès, *Mathematical symbolic systems and functional languages*. Proc. of the «IV Journées Catalanes de Mathématiques Appliquées», 11-13 February 1998, Tarragona (C. Garcia, C. Olivé, M. Sanromà eds.), 175-192.

[5] http://www.mathsformore.com

[6] S. Xambó: *Using* OMEGA *for the effective construction, coding and decoding of block error-correcting codes*. Contributions to Science}, **1** (2), 199-224 (IEC, 2000). A preliminary version appeared in the Proc. of EACA99.
S. Xambó: *Error-Correcting Codes: A computational Primer.* Universitext, Springer-Verlag, 2003 and as a WIRIS-powered book in [5].

[7] S. Xambó, J.M. Miret, X. Hernández: *Completing Hermann Schubert's work on the enumerative geometry of cuspidal cubics in* P^3. The computations in this paper were carried out using the *Omega/Athens* package WIT (see below). This paper was presented by S. Xambó in the Conference in honor of Steven L. Kleiman organized by the Norwegian Academy of Sciences on the occasion of Kleiman's 60th birthday. Has been accepted for publication in the special volume of Communications in Algebra that will be devoted to the Proceedings of the Conference.
S. Xambó: WIT, *a system for computations in intersection theory*. This paper is devoted to present in detail the package the Omega/*Athens* package WIT. In preparation.