APL CAM 16-1, 1994. pp. 50-52

**Letter to the Editor: APL Paleontology Gérard A. Langlet (Feb. 1994).**

Recently, I started a new bibliographic search about ≠ and ≠\ . So, I decided to go back even to the "ancient times" i.e. before scanswere released in APL (in IBM's APL.SV implementation). Then, I could not escape reading again the first "true" reference about APL : K. Iverson's book (1962) that I religiously keep in my private library.

The surprise was great when I discovered p. 252, in the chapter "*The logical calculus*", the properties of the *S* matrix given by relation (7.16). K. Iverson refers to Muller (1954) and to Calingaert (1960). In fact, the "matrix of binomial coefficients" is Pascal's triangle embedded as *P* within a null square matrix; it is known to have, with the same orientation as *S*, e.g. given for $S_3$ in APL notation with index-origin 0, by : `P ←V ∘.! V ← ⍳8` an inverse matrix identical with P in absolute value: `P≡|⌹P` so that, modulo 2, then, for any size, matrix S which is `2|P` must be self-inverse (and so is `⍉S`). Did Ken Iverson feel that S was also the initial of Sierpinski who introduced the fractal construct (first with a curve and not as a binary matrix) in a paper he sent - in French - to the Académie des Sciences of Paris in 1921 ?

Now, this page from "*A Programming Language*" has become for me the most important page of the book, and I do hope it will greatly help people to understand the fundaments of "APL-TOE" [Langlet, 1992], **especially** outside the APL community... , because the language Iverson used in the "ancient times" was readable by professional mathematicians (and still is!).

The properties of the S matrix were introduced in France by Hadamard (the Hn matrices), and used in the US by Walsh together with Rademacher codes - but with the arithmetical negation: In fact, it is a matter of notation:

```
H2 is matrix   |1  1|    H4 is  |H2  H2|     H8 is  |H4  H4|  etc ...
               |l -1|           |H2 -H2|            |H4 -H4|
```

So, every theorem or technique already proved with Hadamard or Walsh transforms also holds, if one now replacesthe arithmetic negation of 1, which returns -1 (noted ¯1 in APL ), by the logical negation of 1 which returns 0. I suspected that the use of 0 (a renormalization in fact) would simplify many algorithms, and I now know it does indeed reduce them to almost nothing.

But the most interesting properties come from the fact that $⌽S_n$ (matrix $S_n$ being noted `S(≠,n)` by Iverson in 1962) is also the modulo-2-matrix-inverse of $⊖S_n$; moreover, such matrices do exist for any order (size), and are cubic roots of a unitary matrix, of course modulo 2; (a self-inverse matrix is a square root of the unitary matrix).

Already, for order 2 (so for shape 22 in *APL*), only

```
|1 1|    and    |0 1|   are non-trivial cubic roots of    |1 0|    the I matrix.
|1 0|           |1 1|                                     |0 1|
```

Since *I* is also its own cubic root, and because the (modulo-2) sum of the three matrices is a ***null matrix***, the S matrix is an exact representation of *j*, the complex cubic root of 1, with four bits only, while it is impossible to represent *j* **exactly** in (IEEE or other) double-precision notation with twice 64 bits for each part, the real part (`¯0.5`) and the imaginary part (`0.5×3*0.5`) in any conventional programming language -

including extended APL dialects which have adopted the `(real)J(imaginary)` convention.

This is a proof that one need <u>not</u> define **first** any negative, rational, irrational number, then constant *i* itself as the imaginary square root of -1, in order to define *j* as a <u>natural constant</u>.

No other number than 0 or 1 is necessary in order to define *j* which has always been for me (as a theoretical crystallographer) the main constant of physics... because *j* is the intrinsic rotation matrix of the Euclidian 3D universe, the space component of Einstein's spacetime; according to Euler, *j* is also the exponential of two thirds of π (exact in extended APL, on the paper only, as: `*○2÷3`).

Then, we may <u>ignore</u> **e** and π as fundamental transcendental numbers: they become <u>consequences</u> of *j* when observations occur (as usual) at a macroscopic scale: all phenomena may be described by secondary laws (Lyapunov exponents, Mandelbrot exponents inter alia) which do not explain the deep TRUE nature of the phenomena: If a phenomenon is, statistically, Gaussian, never forget that the famous bell-shape curve is a) an exponential (squared) construct, b) the envelope of the binomial distribution, then indeed given by the Pascal triangle if the description is discrete and numeric, but by its parity i.e. the Sierpinski *difference-scanned* construct if the description is in spins, genes or parities (there is no other choice than to describe everything with 0s and 1s in computers, even if one does not want to... ).

Now, since $j^2$ is the complex conjugate of *j*, complex conjugation corresponds to a 2nd diagonal symmetry (in APL either `⌽⊖` or `⊖⌽`) ! The choice of any of both matrices, e.g. the "2-geniton G" as done in APL-TOE :

|1 1|<br>|1 0|

as either *j* or $j^2$ is arbitrary. If this matrix is chosen as *j*, then its modulo-2-matrix-square or its modulo-2-matrix-inverse is obtained by `⌽⊖` or `⊖⌽` and equals $j^2$ (and conversely, of course).

The infinite set of symmetric *j* matrices, that one could write (but not obtain on any computer for high values of N), this time, as: `⌽2|V∘.!V←⍳N` in ISO-*APL* is re-written much more obviously as the successive iterates producing - with NEVER any error or truncation - either the rows or the columns of the giant fractal Sierpinskian matrix, with a much smaller ISO-*APL* idiom (the nucleus of every computation as shown [Langlet, 1994] by a recent paper about the Turing machine): `B←≠\B←N↑1` .

This "Bit-Bang" model of our 3D-Universe is, according to the value of N, either large but finite, or infinite..., then either periodical or endless, but <u>by no means</u> reversible, since no minus sign allows anymore to mirror time's arrow in the parity soup of this fantastic **Z/2Z** algebra which <u>ignores</u> cosmological constants, Hamiltonians and equations, as well as a bunch of taken-for-granted concepts; this algebra <u>allows</u>, nevertheless, to rebuild the <u>whole</u> of theoretical and applied mathematics, directly in the native language of the computer. Moreover, `≠\` revisits Shannon's theory, this time with full precision (no smoothing, no averaging) : such a function <u>never</u> produces *noise*, i.e. *entropy growth*, then is the unique possibility to explain and model, without other postulates, how living entities may self-organize and produce such a variety of natural shapes (unexplained by equations) without violation of the 2nd principle of thermodynamics. Sub-periodicities may give the illusion of reversibility. At the quantum level of information processing, then the bit level, <u>micro-reversibility</u> (introduced in 1931 by Onsager, with continuous functions) does correspond to the fact that ≠\1 0 is 1 1 while ≠\1 1 is 1 0 in the same way as ↑↓ and ↑↑ are the pair of possible states for the electron (and other) spins and XY and XX are the pair of possible states for most "living" and "thinking" entities such as you and me.

Many more connections to various fields will be developed at APL 94.

*References*


P. Calingaert, "*Switching function canonical form based on commutative and associative binary operations*". IEEE October Meeting (1960).

L.J. Dickey, "*Gray Codes, Towers of Hanoi, Hamiltonian Path on the N-cube and Chinese Rings*", APL Quote quad (ACM), Vol.24, No2, (Dec. 1993) p. 18-24.

K. E. Iverson, "*A Programming Language*", J. Wiley & Sons, New York, p. 251-5 (1962).

G.A. Langlet, "*Towards the Ultimate APL-TOE*" APL Quote Quad, Vol. 23, No 1, (July 1992) (ACM, APL92, St Petersburg), p. 118-132.

G.A. Langlet, "*Building the APL Atlas of Natural Shapes*", APL Quote Quad, Vol 24, No 1 (Aug. 1993) (ACM, APL93, Toronto).

G.A. Langlet, "*The Ultimate TURING Proof*", Vector (UK) Vol 10, No 3, (Jan. 1994) p. 124-132.

G.A. Langlet "*The Power of Boolean Computing in APL*", accepted for the SHARE Europe Spring Conf., APL Day, La Hulpe, Belgium, (Apr. 19, 1994).

D.E. Muller, "*Application of Boolean Algebra to Switching Circuit Design and to error Detection*", Trans. Inst. of Radio Engineers, New York, EC3 p. 6-12 (1954).


Note. Pr. L.J. Dickey's recent paper (which was also issued in the new APL Canadian magazine "Gimme Arrays") is quoted here:

because... the *Gray code* function is the inverse function of ≠\ (which, iterated, is periodical then also produces the results of its inverse function !);

because a known way of solving Lucas (another French mathematician)' puzzle known as "The Towers of Hanoi" is to use... a Sierpinski triangle (cf. Ian Stewart, Pour la Science (French Ed. of Sc. Amer. ) No 142 p. 103 (1989) already quoted in [Langlet, 1992];

because the "ubiquitous" (as written by Mike Day) ≠\ will solve the cube problem as well as symmetry reduction and ≠ algorithms can solve "*Magic*" - cf. **Vector**, vol.10 No 1, (July 1994), and **Les Nouvelles d'APL** No 8 (May 1994); pure ≠ algorithms were able to solve the "Quinto-Super Puzzle" - cf. **Les Nouvelles d' APL**, Nos 9 & 10 (Oct. & Dec. 1993) ; to be published also in English soon, in 1994, by **APL-CAM Journal**: in such a case, a 0 1 matrix with not less than 8 million items, and with a null determinant, is nevertheless inverted in APL*PLUS II or APL2 (fast APL implementations which do code 0s and 1s in bits and allow large data structures to be processed "not too slowly" by the new "*least-any-power method*" now able to replace the domino which is APL's traditional least-square solver; ⌹ may not be used anymore for large matrices because double precision truncates too many bits, introducing fuzz i.e. **noise**; The "New Mathematics for the Computer" (cf. APL Tool of Thought, New York, Jan. 1993) allow the results not be truncated anymore, in all cases for which exact computing facilities are required : absolute optimizing - a necessity for cryptography and puzzle solving inter alia - is becoming practicable);

because, as now felt then conjectured, ALL puzzles and problems can be solved this way (the most difficult question being to set all the rules of the problems... with 0s and 1s only. APL practice will help more than the practice of any other programming language would, don't you think so?