Building the APL Atlas of Natural Shapes

Gérard A. Langlet
Commissariat à l'Energie Atomique
Direction des Sciences de la Matière /DRECAM/SCM/ Laboratoire d'Informatique Théorique

Centre d'Etudes de Saclay, F-91191-Gif sur Yvette, France
Fax (33) 1 69 08 6640 Tel (33) 1 69 08 65 01

Abstract

It was previously shown [1] that APL contained the most powerful idiom ≠\ that could be used, directly in the language all computers know : binary algebra, to build models as well for physics as for biology and computer science. Several papers on the subject were published or submitted inside the "APL world" as well as outside (Bibliography in [2]). The purpose of the present paper is to show how a classical model, built to generate fractal shapes in plane geometry (2-D) can be revisited and considerably extended, thanks to the properties of ≠\ and of array-oriented binary algebra.

Introduction

The APL-specific ≠\ω idiom gathers two most fundamental concepts of theoretical physics : "parity-symmetry", induced by **1** ≠ ω (equivalent to ~ ω) on one hand, and "parity-asymmetry" induced by "\" on the other hand, which led to suspect that ≠\ω might be considered as a plausible mathematical model of the"Elementary Interaction", usually apprehended by Hamiltonians or tensors in the field of continuous functions. Some mathematical developments indeed proved that a topological parity universe built with ≠\ as its unique force, exhibits properties which are indeed observed and yet unsatisfactorily explained.

During the past year, it was found and published that A) such a topological universe should have *ipso facto* three equivalent spatial dimensions [2], B) that time-irreversibility need not be postulated anymore [3], C) that entropy remains constant although highly-organised structures can appear [4]. It was also found that ≠\ gave the model of the non-integer-order derivation as well as integration, directly at the quantum level of information processing : the bit (see the section involving elementary mathematical theory).

Most shapes in Nature are now recognised as fractals, from snowflakes to galaxies, and ≠\ alone indeed allows to write the most condensed program to generate a fractal, the Sierpinski gasket, as well as discrete holograms [1,2]. But another question was : Could ≠\ also generate the shapes we observe everyday around us, flowers in the gardens, crystalline, molecular and biological structures seen under the microscope or studied by spectroscopy, patterns painted, carved or woven by artists who ignore mathematics, physics and biology ?

Of course, such a question is ambitious... but not irrelevant. At the present time, one cannot answer it because not enough people have worked on the subject (and because too many scientists still ignore APL).

But many mathematicians, botanists, physicists, artists and philosophers did study and observe shapes in the past, from the Greek to Leonardo da Vinci, from Church to Mandelbrot, from Cantor to Julia, Agosti, Barnsley, Valéry and Thom. One of the most simple and clever approaches in plane geometry was proposed in 1903 by von Koch in order to explain the shapes of snowflakes. At the same time, the model helped people to discover that the length of a coastline depends from the elementary quantum, i.e. from the unit that the geometer uses when he measures the length, step after step.

The following sections will first describe an APL-function "SHAPES" which extends von Koch's model into various directions. Many natural shapes already indeed appear in a two-dimensional environment; all shapes are "strange attractors", usually modelled by continuous functions, with some mathematical difficulty, many restrictions and, in general, some high budget for computer experiment.

This is just the beginning of the APL-Atlas which might be extended to 3-D objects in the future. May this function bring the proofs :

1) that APL is the most powerful tool to study shapes with a new eye, the one of pure binary algebra (the quantum field of information) as much as possible;

2) that indeed ≠\ alone produces a variety of shapes which look familiar, sometimes strange but always aesthetic, close to flowers and to molecular structures; these latter cannot be thought of easily, if one tries to forge continuous functions for that purpose; for mathematicians, physicists, biologists, which were interviewed, there is no hope of finding one day a continuous function which can describe a human being... while man is indeed described by 23 nice pairs of chromosomes which represent a large but finite number of *bits*; then, some simple binary mechanism must BE *the* program;

3) that no high budget is anymore necessary to obtain fractals on microcomputers even when complex algebra is not available.

The SHAPES function

Here is function SHAPES which produces 2-D shapes (arguments and sub-functions will be explained in detail afterwards. Function EXAMPLES, in the Appendix, proposes a selection of arguments which generate nice patterns). "Opt." means "default option".

```
     ∇ S SHAPES N;B;D;I;K;L;M;⎕IO
[1]  L←0=⎕NC 'S' ◇ ⍎L/'S←0' ◇ B←,⎕IO←0                        ⍝ S : Symmetry {6}
[2]  D←÷2 ◇ T←2 4⍴0 1 0 1 1 1 0 1                             ⍝ N : Recursivity, {ID No}
[3]  'B←,T[B;]' do | 1↑N ◇ I←⌈D×⍴B ◇ B←I↑B                     ⍝ Contour bit string B
[4]  L←0<K←1↑1↓S ◇ S←1↓S ◇ ⍎L/'B←COG B'                       ⍝ Opt. Cognitive transform
[5]  L←0>K ◇ S←S+6×0=S ◇ ⍎L/'B←HEL B'                         ⍝ Opt. Helical    transform
[6]  I←S,⍴K←M←2 2⍴M,⌽1 ¯1×M←2 1○○2÷S◇ M←I⍴M                  ⍝ Rotation Matrix : 2 2≡⍴M
[7]  I←0 ◇ T←K ◇ 'M[I←I+1;;]←T←T+.×K'do S-1                   ⍝ M: S first powers of M
[8]  'B←≠\B' do 1↑1↓N ◇ K←S×⍴B                                ⍝ Opt. binary integration
[9]  K←K×1+I←≠/K⍴B                                             ⍝ Parity of B must be even
[10] K←⌽2,⌊K+D←÷2 ◇ M←K⍴M[;;0] ◇ I←1+2↑⌊S×D                   ⍝ Matrix indexing
[11] T←S| ¯1+ ¯1⌽+\I[K[0]⍴B] ◇ B←M←K←⍴T←M[T;]                  ⍝ Relative coordinates : T
[12] 'T←T×K⍴2/1,∆g+GOLD×∆g∧.=0'do 2=⎕NC '∆g'                    ⍝ Opt. irregularity
[13] T←+⍀T ◇ T←T×I←255 ◇ T←D+T ◇ K←⍴T←⌊T                      ⍝ Absolute coordinates (for absolute plotters)
[14] T←T-K⍴⌊⌿T ◇ K←I÷⌈/T←,T ◇ T←T×K ◇ T←T+D                   ⍝ Scale adjustment (especially to compact results)
[15] T←⌊T ◇ REDUTN ◇ T←T,2↑T ◇ PLOTT                          ⍝ Opt. Compaction and Plot (implementation-dependent)
     ∇
```

The result T is a global vector X0 Y0, X1 Y1,.... Xn Yn, X0 Y0 here containing integers from 0 to 255, so that it may be kept as a character vector, using a converter such as T←⎕AF T in APL2. Due to the fact that T←+⍀T is the unique arithmetical operation (line [13]) performed on coordinates, precision on the absolute coordinates thus obtained is at its best, even on sophisticated shapes; then, enlargements of small sub-images can be easily obtained. Of course, constant 255 may be increased in order to keep T as an array of integers with more precision, e.g. to 1023 or 65535; however, this is almost useless on common CRT's (e.g. PC, Mac, Atari).

GOLD is the golden section 1.618033988749895, either a variable or the niladic function

```
    ∇ R←GOLD
[1]   R←.5×1+5*.5
    ∇
```

Subfunctions :

∆E is a character vector, an APL expression. ∆E **do** N shall iterate N times expression ∆E .

C←COG B returns C the "Cognitive Transform" of B.

H←HEL B returns H the "Helical Transform" of B.

(B is a bit-vector. See the cognitive-transform paragraph.) The binary matrix operators which can perform the transforms were named "Genitons" [1]. The 2- and 4-genitons are:

```
                1 1 1 1
1 1             1 0 1 0
1 0             1 1 0 0
                1 0 0 0
```

If G is a conforming geniton (a square Sierpiński matrix, so that 1↑⍴G is the same power of 2 as Q i.e. ⍴B), then H is given by ⌽G≠.∧ B and C by G≠.∧⌽B .

However, much faster algorithms, with execution time proportional to Q, were described elsewhere [5]. If E is the elementary sequence E←Q↑1 , then the Q successive rows of matrix G are obtained iterating Q times E←≠\E . Then, the last row, as well as the last column of G (which is a symmetric binary matrix), reproduces the original E. Here is the 4-geniton (when Q is 4):

```
1 1 1 1  Every (2k)-geniton contains three k-genitons
1 0 1 0           { North-West }
1 1 0 0  as its   { North-East }   quadrants :    1 1
1 0 0 0           { South-West }                  1 0   .
```

The South-East quadrant always contains 0's only.

Note the identity (for ⍴B a power of 2):

(⌽ COG B) ≡ HEL ⌽B .

REDUTN reduces the size of T, detecting repetitions; e.g. if the first half of T is the same as the right half, T gets reduced in length by 2, recursively. This is also applied to thirds, fifths, etc... at will. The use of REDUTN is optional; the main advantage comes from the possibility of keeping the graphic results as character vectors, then in compact APL-independent system files.

PLOTT (implementation-dependent) has to plot the graph of T. Every odd item in T is an abscissa, every even one an ordinate, so that any "polyline" software is convenient. In (APL*PLUS II), colour patterns can be generated as a function of S, the basic symmetry; the number of vectors may exceed 100,000 with no difficulty (with large S or with recursion 7).

When T does not refer to a periodical shape (last point different from the first one), PLOTT shall detect this and remove the two last items of T which were appended by T←T, 2↑T .

More comments

When the left argument S (for symmetry) is undefined, 6 (hexagonal symmetry) becomes the default option, so that SHAPES 4 will produce von Koch's snowflake at the 4th order of recursion, which is enough for most screens of microcomputers. When recursion increases by 1, the number of points increases by 4:

As in the snowflake, every segment is replaced, at every recursion, by 4 segments so that:

 becomes  ,

then the number of bits which is necessary to describe a closed contour by a binary vector is:

2*0⌈¯1+2×R←1↑N   with R the order of recursion.

Note: Although the word "recursion" is kept in the text, the SHAPES function is not recursive (so as to improve speed and decrease memory use).

R≡0 corresponds to a polygon (starred if R is odd and ≥5). (All rotations between two consecutive segments are left turns).

R≡1 corresponds, for S≡6, to the bit vector 0 1, which will be repeated as much as necessary to close the contour (the David star has 12 segments with alternate left and right turns).

A left turn will be with angle α ○2÷S i.e. in this case 60°. A right turn will be with angle −α i.e. in this case −60°. S≡6 and B≡0 1 completely describe this concave polygon.

R≡2 will correspond to sequence 0 1 0 1 1 1 0 1 i.e. to the former sequence 0 1 in which 0 is replaced by 0 1 0 1 while 1 is replaced by 1 1 0 1.

R≡3 corresponds to the 32-bit sequence B:

```
0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1
{   0   }{   1   }{   0   }{   1   }{   1   }{   1   }{   0   }{   1   }
```

etc..., with the same recursive replacement.

In SHAPES, sequence B is obtained at the end of line [3]. For R≡7, it has 8192 bits; it is then repeated to form the full contour of a von Koch's snowflake (when S≡6) which has 12 times as many very small elementary segments, i.e. 98184. So, a 1-kilobyte binary sequence codes this jagged fractal.

Note. From R≡0 to R≡1, the replacement of 0 by 0 1 0 1 produces... 0 1 0 1 which is periodical and, then, can be reduced to 0 1 only.

Towards many more shapes

Until now, what has been explained about SHAPES is classical. Von Koch imagined his construct at the beginning of this century, in order to understand real snowflakes he could see on his windows. In fact, all snowflakes are different from one another. Many chemical compounds (metal salts) also crystallise on flat surfaces giving very beautiful symmetric and asymmetric fractal patterns. With the help of APL, and of the generalised theory of binary integration (ref. [6], [7], etc...), we have tried to extend the classical von Koch's construct:

1) to any symmetry,
2) to irregular shapes,
3) to any integral,
4) to their transforms :
   cognitive (C) and helical (H);

the goal was to gather all these extensions into only one small function which should execute very quickly in any common APL implementation available on regular microcomputers. This is the reason why ISO-APL is used here (the only non-ISO-APL extension is "replicate" which is now accepted by all interpreters).

Especially for high orders of recursion, when B is replaced by one of its ⍴,B successive integro-differentials, the number of possible shapes will increase. If B is replaced either by its cognitive transform C, or by its helical transform H, all the successive integro-differentials of C and H are also different from B and from its integro-differentials (in the general case). Then, the following table gives the maximum number NS of different shapes that are expected, i.e. 3 times ⍴,B at every order of recursion, for every possible symmetry :

R	: 0	1	2	3	4	5	6	7
⍴, B	: 1	2	8	32	128	512	2048	8192
NS	: 3	6	24	96	384	1536	6144	24576

In addition, line [12] of SHAPES allows irregularity. The global variable ⍙g may contain any numeric or Boolean vector. This line is not executed when ⍙g is absent. When ⍙g is 0, the length of all successive segments becomes modulated by 1,GOLD repeated all along the contour. With ⍙g≡10 100 1000, the successive length of segments will be modulated by 1 10 100 1000. It is possible to include negative values. So, "random" fractals, such as imaginary islands are easily composed by SHAPES. But the best results were NOT obtained by this technique... which is, in fact, "ad hoc".

The nicest displays come from studies in symmetries that are not commonly investigated; as an example, symmetry 30 conjugates symmetry 6 (frequent in chemistry : graphite, benzene molecule) and symmetry 5 (the one of quasi-crystals). Such a conjunction is common in living structures (eggs, viruses, the guanine and cytosine bases of DNA and RNA, which exhibit a 5-atom ring, adjacent to a 6-atom ring). It forms the quasi-spherical structure of the "fullerene" or "footballene" C_{60}, new variety of Carbon - look at a soccer-ball... So, exploring symmetry 30, one could expect some surprises. This happened to a biologist, M. V. Locquin, SHAPES on Atari ST (APL.68000), with 30 SHAPES 5 21 , which produces in 7680 points an astonishing pattern of 15 bees or flies, linked by magnified fly-eyes...

A glance to the mathematical theory and physics

As written in previous papers, all the main laws of physics were obtained by Newton, Laplace, Maxwell, Navier-Stokes, Schrödinger, Kortweg-de Vries, Sine-Gordon and Einstein, among many physicists and mathematicians, using integro-differential equations.

In a variety of books about fractal geometry and natural-shape synthesis, some mentions concern the Riemann-Liouville integral. This mathematical construct is, at the present time, the best tool which allows to build the so-called "non-integer integrals" and the "non-integer derivatives" of a continuous function. The main use of it was to simulate the visco-elastic behaviour of matter (lava, bread and pizza paste, marshmallow, rubber, fused glass, emulsions, polymers etc...), which is very important for almost all modern industries.

Other names of such mathematical tools are "fractional", "half- integer" integration and derivation operators.

First, any signal, any variation of any continuous function can be described by a convenient sequence of bits. A good example of this is digital recording (CD), which has reached the HiFi quality in the recent years. So, the 5th Symphony by Beethoven (or Mahler) can be expressed as a binary sequence B (equivalent to a modulation in frequency).

Second, APL contains THE MAGIC IDIOM ≠\ which corresponds, for bit sequences, exactly to what Riemann-Liouville integrations and differentiations do with continuous functions. Moreover, the choice of the coefficients α and $(1-\alpha)$ for interpolation, i.e. mixing a function and its integral or derivative, is left to the user : this latter chooses a value for α which fits at best his experimental results. But, the formulae for the "half-integral" and the "half-derivative" are unfortunately NOT so simple with the Riemann-Liouville theory. Analytical feasibility is reduced to some special cases...

Discrete differentiation for small (but not infinitely small) intervals Δt on a function $F(t)$ will give the variation of a studied phenomenon $[F(t+\Delta t)-F(t)]/\Delta t$ as a vector, with the length of the initial data minus 1, resulting from sampling (experimental measurement) e.g. at $\Delta\cdot t = 0, 1, 2, 3,$ etc...

Transposed to Modulo-2 algebra, then to the binary field, the arithmetic difference becomes MINUS Modulo 2, then the EXCLUSIVE OR, then the logical difference, i.e. simply ≠ in APL. The first binary difference on any binary vector B is :

```
1↓B≠¯1⌽B
```

In general, in physics, $F(t)$ is unknown. Most perceptions and measurements correspond to differences along axes of space or time. So, one sets, in the most common case, some differential equations. The goal consists in finding the solution(s) for F from these equations. For most natural phenomena, it is a hard job. Fortunately, with the help of the computer, one performs discrete integration. In APL, discrete integration (undefined integral or CUMULATION) is simply performed by `+\` (while the defined integral or SUM is `+/`) . Transposed to Modulo-2 algebra, then to the binary field, the arithmetic sum becomes PLUS Modulo 2, then the EXCLUSIVE OR, then the logical sum, i.e. simply ≠ in APL, another time.

Then, since ≠ keeps all the properties of + (associativity, commutativity), binary algebra becomes the UNIQUE case for which the difference is also associative and commutative.

The propagated difference or differential of B is `≠\B` while the local differential will be `B[⎕IO],1↓B≠¯1⌽B` for any vector B. The interest of correcting the first item is that :

1) the shape of B and the one of its local differential remain the same;
2) the local differential is identified with the Gray code of B (cf. [8]);
3) the following identity is true for any vector B :

```
B ≡ ≠\ B[⎕IO], 1↓B≠ ¯1⌽B
```

Any binary integral (undefined integral of any signal B) is given by `≠\` as well as any propagated differential : All the mathematical difficulties, inherent to the use of the Riemann-Liouville functions, disappear. `≠\` has "something of an integrator" and "something of a differentiator". We shall call it the unified "Integro-Differentiator" (ID for short).

APL is the *unique* programming language with an ISO standard [9], that can offer, as a convenient notation which extends to arrays, i.e. "parallel vectors", THE tool that all physicists should know before thinking of their physical problems and tring to model fractality or chaos with a computer.

So, we had the feeling - and any physicist can check this immediately, if he knows APL - that mathematics, applied to physics, might be drastically simplified, using this new method of investigation, able to mimic the elementary interaction, if not to describe it completely at the level of fundamental parities i.e. the information quantum either 0 or 1, cf. [10].

Moreover, `≠\` may be applied as well as its inverse function, as many times as one wishes, always with no error. Every signal, then every discrete function, with a large number of so small intervals that the plot looks continuous, could become a priori infinitely "non-integer integro- differentiable", and, last but not least, easily.

A rapid study of the properties of `≠\` iterated, let appear not only confirmations, but also unknown properties of Boolean algebra, with fantastic progresses about periodicity, discrete holography, genetic automata, symmetries, chirality (the left/right or "levo/dextro" asymmetry), neuronics, wave propagation and... applied mathematics.

If the hypotheses of physicists are correct, given any system, described by a function such as $F(t)$, the status of the system after Δt (e.g. a small unit interval of time, a "quantum"), is given by $F'(t).\Delta t$ which becomes $F'(t)$ if Δt is a unit interval. For many "independent" variables, one uses total differentials; before understanding complex systems, let us first try to discover what happens to a signal, with just one variable. Physicists suspected that fractality required non-integer integro-differential formulae which are not currently taught at school.

So, according to theoretical grounds for signal sampling, i.e. measurement, which lie beyond the scope of the present paper, and after exploring the properties of array-binary algebra, we came to the following hypothesis :

Given any binary sequence B which describes e.g. any signal as a function of some parameter, its "next future" should be given by `≠\B` , moreover with NO ERROR because ≠ is a logical function.

This hypothesis was checked on several systems, using 0 and 1 to describe parities : less-stable/more-stable, absent/present, massless/weighing, anion/cation, cold/hot, fundamental/excited, recessive/dominant (in genetics), and seemed to lead to very good results : in fact, no exception to the rule was found...

A very simple example in the field of computer science is the MS-DOS command COPY. If we want to copy a public-domain APL from disk A: to disk B: , we may write : COPY A:APL.EXE B:

If 1 is the existence of file APL.EXE on the diskettes inserted into the drives A: and B: of a PC, the initial state before copy is one of the 4 following ones :

	A: B:	A: B:	A: B:	A: B:
	0 0	0 1	1 0	1 1
Future forecast is `≠\` applied to these vectors i.e.	0 0	0 1	1 1	1 0 .

We want to propagate APL.EXE from A: to B: If the file does not exist on A: , is the final state the same as the initial one? The answer is : "TRUE".

If the file exists on A:, not on B: , the subsequent state is 1 1 (successful COPY). So, the hypothesis is TRUE.

If the file also exists on B: (fourth case), the subsequent state is NOT the one after a copy : we have a conflict. ≠\ immediately reveals a necessary condition : the existing file on B: with the same name (although the content may be different) MUST be deleted first (anti-copy). So, there is a new subsequent state 1 0, which is the same as the third initial state. Now, the copy can take place normally; (some systems ask the user about his intentions before destroying APL.EXE from drive B: while MS-DOS does not). ≠\ detects that there are two steps in this last situation.

≠\ gives, on a very long sequence, the possibility of obtaining by successive iterations what we call the successive ID's (for Integro-Differentials) or QUID's (for "Quantum Unified ID" in papers for physics, because one can study models at any desired scale of quantisation). This has been studied with APL for some years now and published [1], [2], [5]. When the length of B is c a power of 2, the cth ID always reproduces B. For any sequence B, ≠/B (by analogy with +/) is its defined integral. It is 1 if the sequence has an odd number of 1-bits, thus expressing the parity of the whole sequence. ≠/B has the same value as ¯1↑≠\B except that the latter idiom produces a vector with length 1 instead of a scalar.

The vector containing the last item of each successive ID of B was named C for "Cogniton". C is the cognitive transform of B. It contains :

```
        (≠/B),(≠/≠\B),(≠/≠\≠\B),(≠/≠\≠\≠\B),
etc...    or

∊¯1↑¨(≠\B) (≠\≠\B) (≠\≠\≠\B) (≠\≠\≠\≠\B)
etc...  in APL2.
```

This transformation is an involution (just like the Fourier transform, but always exact), so that the cogniton of C is the original B (we restrict here to an integer power of 2 for ⍴B).

The other transformation which is used in SHAPES corresponds to the helical transform H. The mirror of H, i.e. ⌽H contains :

```
(≠/B),(≠/≠\¯1↓B),(≠/≠\≠\¯2↓B),(≠/≠\≠\≠\¯3↓B),
etc...
```

The helical transformation is also an involution : the helical transform of H is the original B, and ⌽H is also the cognitive transform of ⌽B (see [1, 2]).

All these properties in vector and matrix binary algebra (a field which seems to have never been explored systematically in mathematics and in computer science) were discovered thanks to APL, and may be used directly in the memory of the computer, on giant arrays, thus short-circuiting huge computer-business and most frequently-used methods of applied mathematics...

But what has now become possible, thinking in APL and using it, has probably be used by Nature itself before APL was invented... (The least-action principle does not state anything else.)

As an attempt to prove it (many other directions are explored in parallel), the SHAPES function was written. The connection with fractals, namely with Sierpinski's work, was shown before [1, 11]; a new idea was to use the Sierpinski construct as a matrix operator so as to replace the Fourier transform by something much more simple, efficient and accurate.

Genetic automata (see [2], [12], [13]), based exclusively on the ≠\ rule, immediately led to a new explanation of Mendel laws without any ad-hoc hypothesis. Similarly, ≠\ can also produce and explain the famous "1/f-signals" which had no satisfactory mathematical model in physics [2], [6].

Towards natural shapes

The most useful mathematical construct used in research for the synthesis of fractal (i.e. realistic) images is indeed the Fast Fourier Transform (FFT) [14] (which is also connected to the Riemann-Liouville integral).

Then, the idea that Nature could also use some simple mechanism, based on parity hierarchies, and very close to the cognitive-helical transforms to build its complex shapes, was in the air. We also knew that ≠\ was intimately connected to the Fibonacci series, which produces the parastichies of some flowers (sun-flower), the shapes of pine-cones and pine-apples, with visible orthogonal spirals which can be counted, and sometimes also modelled in fluid dynamics [15] [16].

But "what can be counted" does not cover all shapes (unfortunately named "random" fractals because they were not understood).

Most flowers do not exhibit Fibonaccian structures (dahlias do, roses, apparently do not). In two dimensions, or almost-2D, Nature produces creeping plants, (ivy, honeysuckle) as well as naphtalene, anthracene, graphite (these compounds are formed in a plane or, for graphite, built by stacking of planes of carbon atoms with hexagonal symmetry), and frost crystals on windows for Xmas. Could only one mechanism lead to such a variety of shapes ?

Von Koch's initial idea was excellent (simple ideas are always good). It also led to explain the coast-line length paradox : the length of a ragged coast depends on the scale at which the measurements are performed. Every book on fractals, chaos or mathematical games does show this construct. Nature is indeed fractal. So is ≠\ which could be used in order to improve, as explained above, the variety of accessible shapes, without much numeric computation, except in order to remain compatible... with existing graphic software and habits.

This is just what we have done (limiting ourselves to 2-D experiments so that microcomputers with APL may be used).

The first goal was to avoid recursivity and to accelerate every sub-algorithm using a minimum of arithmetics or trigonometry.

The second goal was to minimise errors, another good reason to work with bits.

So, given a symmetry, e.g. 6-fold (hexagonal) symmetry to start with, the one of snowflakes and of honeycombs, we just need to compute once ONE rotation matrix; for the 6-fold symmetry, ω is 60°:

$$\begin{matrix} \cos\omega & \sin\omega \\ -\sin\omega & \cos\omega \end{matrix}$$

Then, around the central point, we may turn S times; for S≡6, we can generate the other matrices, for 2ω, 3ω, etc... raising this matrix to its successive powers (line [7] of SHAPES). The Sth power of such matrices has to be a unit-matrix. In fact, most lines of SHAPES could be used only once : Sequences for recursions lower than 7 are the first items of the sequence for 7. Cognitive and helical transforms also have such properties : C for R≡5 is ¯512↑C for R≡6 , while H for R≡5 is 512↑H for R≡6 , etc...

The necessary successive powers of the rotation matrices for significant symmetries (4, 5, 6, 8, 12...) can also be computed once and kept in a file or in the workspace.

Line [8] is not optimal: it iterates B ← ≠\B 1000 times if you want the 1000th ID. This expression is just for demonstration; otherwise, optimal algorithms can give you the 1000th ID directly, without computing the 999 first ones, see [7].

A turtle at work

As far as rotation matrices are concerned, we will only use one half of them; this explains the M[;;0] of line [10]. (M[;;1] is useless) : in order to plot the shape, the graphic cursor operates exactly like the turtle of LOGO. Every displacement is possible in one of the S centro-symmetric directions around the origin, e. g. the center of the screen : for S≡4, the turtle may go East, North, West or South as far as one quantum at each move. But when the quantum is small (lower than the physical pixel), the turtle can reach any point of the screen, nevertheless, cumulating its tiny moves.

For a displacement with angle ω , X is cos ω and Y is sin ω. These values are exactly the 2nd column of each matrix, so that no computation is anymore necessary; indexing is enough.

The role of the turtle (it does so when R, recursion, is 0 and B also 0), is to start from the origin, go a quantum along one direction, turn left by ○2÷S (60° when S is 6), go a quantum that way, turn again left by the same angle, go another quantum that way, etc... until it gets back to the origin. On some CRT's, the Oy-axis goes down so that the image may be reversed; this has no importance at all for the shape. Then, {6} SHAPES 0 will produce an equilateral triangle, not a hexagon. Although the scale is the quantum, the automatic centering and resizing in lines [13] and [14] will magnify this triangle.

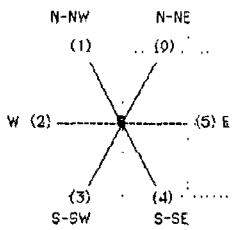
For R≡1, then, with B≡0 1 , the turtle becomes more subtle : From the origin, it still goes along one direction for one quantum, turns left by ○2÷S because the first item of B is 0, goes a quantum that way and turns left by ○¯2÷S , i.e. right. by the absolute value of the same angle, because of the 2nd item of B, which is 1. Sequence B is the turtle's computer program, its Turing ribbon, which is repeated as much as necessary.

The first 0-turn applies power 1 of the rotation matrix, i.e. cumulates the second column (X and Y) of the rotation matrix itself with its initial move (X0, Y0) before turning.

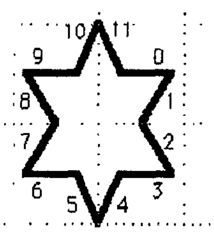
Before line [11] is executed, for SHAPES 1, symmetry S is 6, the binary sequence B is 0 1 and I is 4 1 (given by 1+2↑⌊S÷2) at the end of line [10]. K[0] is 12, i.e. the pre-computed number of segments forming David's star.

The APL expression : S|¯1+¯1⌽+\I[K[0]⍴B] produces : 5 3 4 2 3 1 2 0 1 5 0 4 which is put into variable T.

This vector of integers contains (this will hold for any S and R) directly the indices in 0-origin, of the successive directions taken by the subtle turtle : it goes East (5) then South-SouthWest (3), then SE-SE(4), then West (2), etc... until the last move S-SE (4) which brings it back to the initial location, closing the contour. The successive moves are numbered from 0 to 11 in the following schema.



M already contains the elementary X and Y for the S possible rotations. T←M[T;] recreates T as the array containing the incremental moves both along X and Y for incremental plotters; this has been obtained with no arithmetics or trigonometry once the 2nd columns of the successive S powers of the suitable rotation matrix are known. The absolute coordinates of the successive turtle locations are given by +\T and will fit CRT graphics (⎕GLINE, vdi or cgi "polylines", etc...).



Roundings are done after obtaining the absolute coordinates, so that errors remain, even for many points, unsignificant.

How a turtle draws flowers and natural shapes

Extending the snowflake construct to all symmetries and to all ID's of cognitive and helical transforms was just a small idea that could a priori prove to physicists, botanists and crystallographers, that Nature knew ≠\ before the "scan operator" was introduced into APL.SV a decade and a half ago (and then IS an ideal and general plausible mathematical model of the Elementary Interaction). Although the SHAPES function restricts to 2-D graphics, the surprise was great when the results came out, especially in full colour. Function EXAMPLES contains some suggestions, but experiments with ≠\ are still a virgin island... (The whole documented software will be available as a small workspace for APL*PLUS I & II, easily adaptable for any other APL-implementation, at the Software Exchange booth). The plates, unfortunately in black-&-white, give a faint idea of what shapes APL can produce at low cost.

Plates Each plate was obtained in a few seconds on the Atari ST, a slow machine at 8 MHz, using APL.68000, with either a newly-written polyline/polymarker function, or direct adressing of the graphic screen as a bitmap (just the equivalent of a 256,000-bit APL vector) which shortcuts all the previously-necessary graphic software and may be extended to produce also colour output.

References

[1] G.A. Langlet, Unitary Theory of Information, APL-CAM Journal, Belgium, Vol 13, No 2, p. 399-432 (Part I) & No 3, p. 709-743 (Part II) (1991).

[2] G.A. Langlet, Towards the Ultimate APL-TOE, APL92, St Petersburg; APL Quote Quad, ACM, SIGAPL, USA, Vol. 23, No 1, p. 118-134 (July 1992) and references therein.

[3] G.A. Langlet, L'Intégration Binaire, Clé de l'Irréversibilité du Temps, 13th Int. Congress on Cybernetics, Namur, Belgium, (Aug. 1992).

[4] G.A. Langlet, Physique et Algèbre de Boole, Les Nouvelles d'APL, AFAPL, Paris, No 3, (April 1992).

[5] G.A. Langlet, Un Concours Paritonesque, APL-CAM Journal, Belgium, Vol. 14, No 3, p. 412-416 (July 1992).

[6] G.A. Langlet, La Propagation Asymétrique de la Parité Biomath, ISSN 0035.10.24, vol. 115, p. 5-42 (1992).

[7] G.A. Langlet, L'Intégration Binaire par l'Image, APL-CAM Journal, Vol. 14, No 3. p. 389-406 (July 1992).

[8] H.A. Peele, Gray Codes in APL, APL Quote-Quad, Vol. 7, No 3 (1976, revised, priv. comm.).

[9] ISO8495e/f, Int. APL Standard, ISO, CH-Genève (1989).

[10] G.A. Langlet, Le Principe de Moindre Action Généralisée, UITF 1992 Rennes, ISBN 2-9009947-00-9 p. 77-96 (Dec. 1992).

[11] G.A. Langlet, Variations sur Sierpinski Pascal & Fibonacci, APL-CAM Journal, Vol 13, No 2, p. 375-389 (1991).

[12] G.A. Langlet, The Fractal Laws of Genetics, Biomath, ISSN 0035.10.24, vol. 118, p. 60-71 (1992).

[13] G.A. Langlet, DNA-style Programming for Genetic Simulations ,APL-CAM Journal, Vol. 14, No 3, p. 407-411 (July 1992).

[14] R.F. Voss, in The Science of Fractal Images (Peitgen & Saupe, eds), Springer, NY, ISBN 0-387-96608-0 (1988).

[15] R.V. Jean, On the Origins of Spiral Symmetry in Plants , in : I. Argittai & C.A. Pickover (eds), Spiral Symmetry, World Scientific, Singapore, ISBN 981-02-0615-1 (1992) p. 323-351.

[16] R.V. Jean, Mathematical Approach to Pattern and Form in Plant Growth, J. Wiley & Sons, New York, USA, ISBN 0-471-88357-3 (1984); Growth Matrix, p. 144.

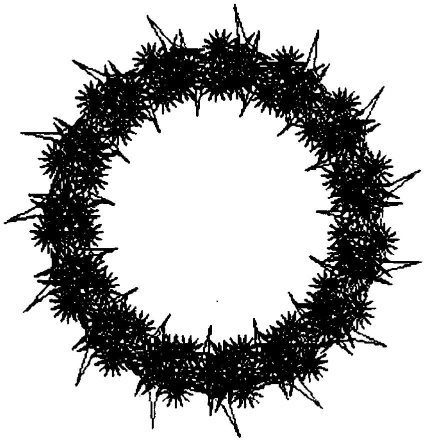
[17] J.A. Brown, S. Pakin & R.P. Polivka, APL2 at a Glance, Prentice Hall, N.J., USA, ISBN 0-13-038670-7 (1988);

[18] G.A. Langlet, New Mathematics for the Computer, Tool of Thought, ACM/SIGAPL, (New York, NY, Jan. 23, 1993).

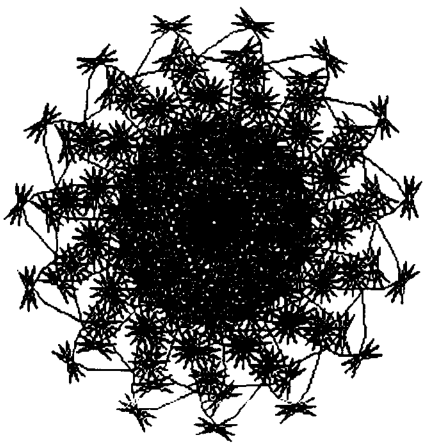
[19] S. Wolfram, Theory and Applications of Cellular Automata, World Scientific, ISBN 9971-50-123-6, p. 11, 52, 131 (1986).

EXAMPLES

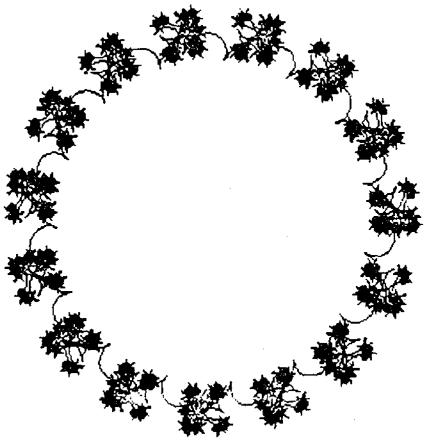
```
                                                  ⍝ { } default option

  9 ¯1  SHAPES 5  89                              ⍝ Nice flowers
⍝ ↑  ↑           ↑  ↑__ Integro-Differential (ID) number
⍝ ↑  ↑           ↑_____ Order of Recursion  (from 0  to 6 or 7 with high ⎕WA)
⍝ ↑  ↑_________________ Transform.  ¯1 : Helical Transform,   {0} : No Transform,    1 : Cognitive Transform.
                                                  ⍝ {6 0}  is taken with monadic syntax.
⍝ ↑____________________ Symmetry (3 or 4 or 5 or 6 or 8 or 9 or more...)  Reduce recursion for high symmetry codes.
        SHAPES 5                                  ⍝ Von Koch's snowflake (symmetry 6,  recursion 5)
  5     SHAPES 6                                  ⍝ Pentagonal snowflake
  5     SHAPES 5  22                              ⍝ Celtic ladies
  5  1  SHAPES 6  11                              ⍝ Strange runner
 15 ¯1  SHAPES 6 732                              ⍝ Chestnut crown
 30     SHAPES 5  21                              ⍝ Fly crown
 32     SHAPES 5  71                              ⍝ Handsome pilot-fish
104  1  SHAPES 4  31                              ⍝ Close the apple of the eye,  increasing ID from 31 to 63 then 127;
                                                  ⍝  also  try  ¯1  instead of 1    for  less aperture  until shutdown.

⍝  More to try (with high recursion-orders, with APL⋆PLUS II...)
⍝  Indications for nice colour display is available.
⍝  Colour output will be included in the versions for the software exchange.
```



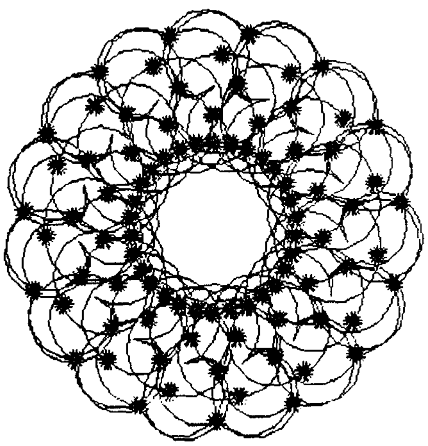
"The Chestnut CROWN"
34th ID of the
cognitive transform
for Recursion 5
in 15-fold symmetry



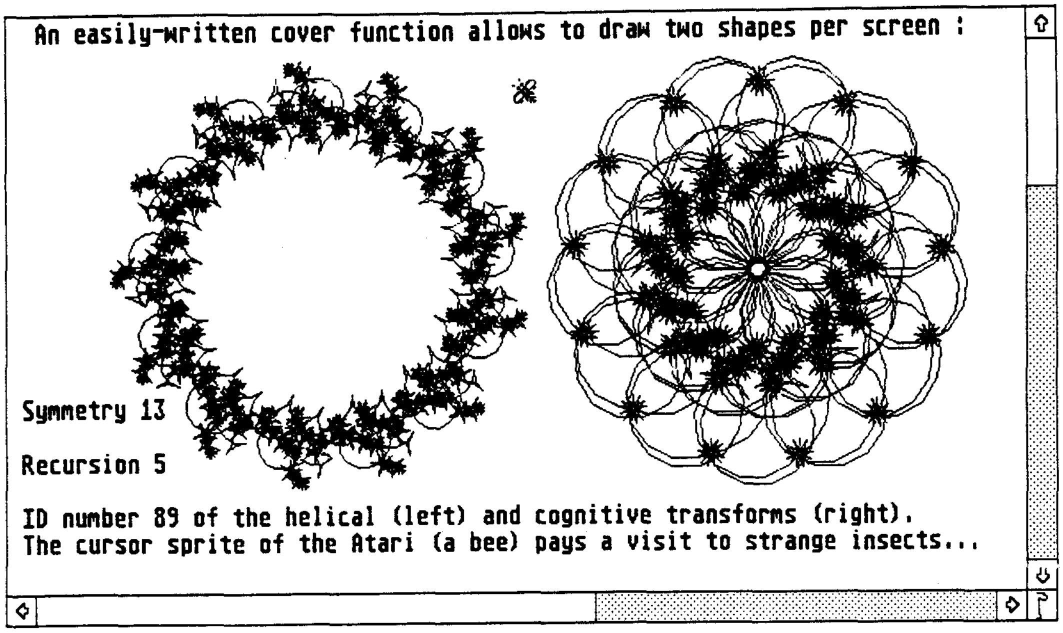
Asymmetric rosace flower :
55th ID of the cognitive
transform for Recursion 5
in 15-fold symmetry.
Interest lies not only in
the final output, but also
in the way it is plotted.

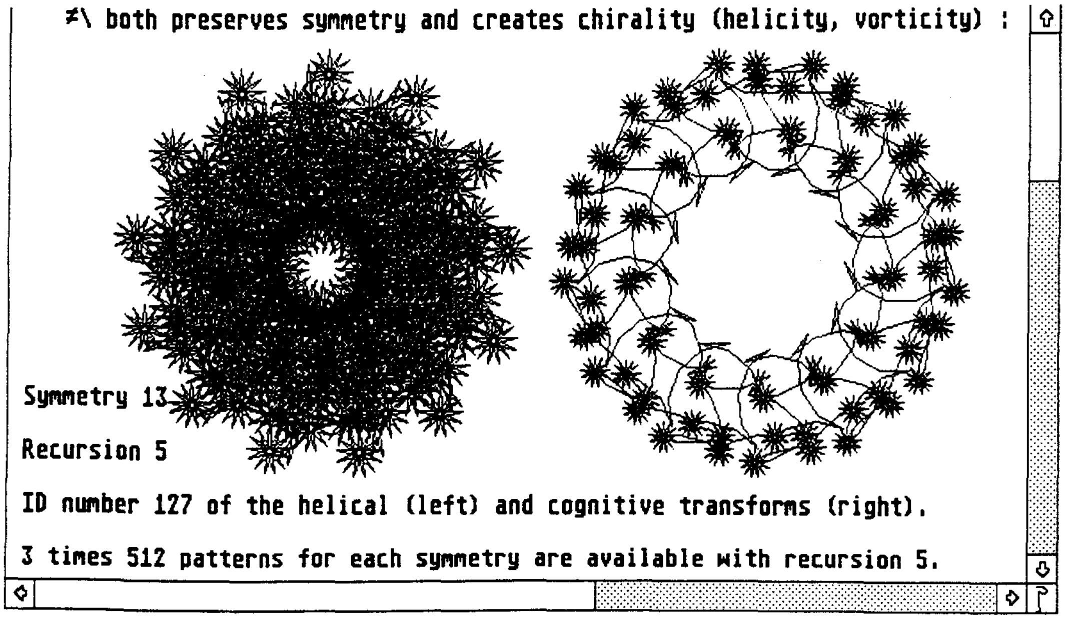


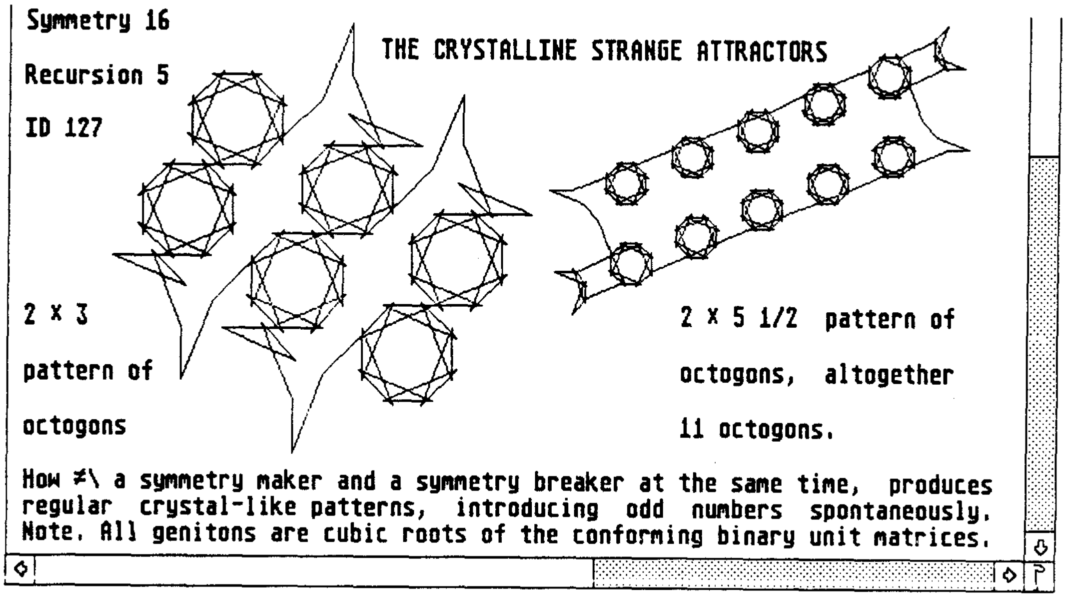
The 89th ID of the helical transform for Recursion 5 in 15-fold symmetry builds a completely different bunch of attractors (splendid when colour may be used).

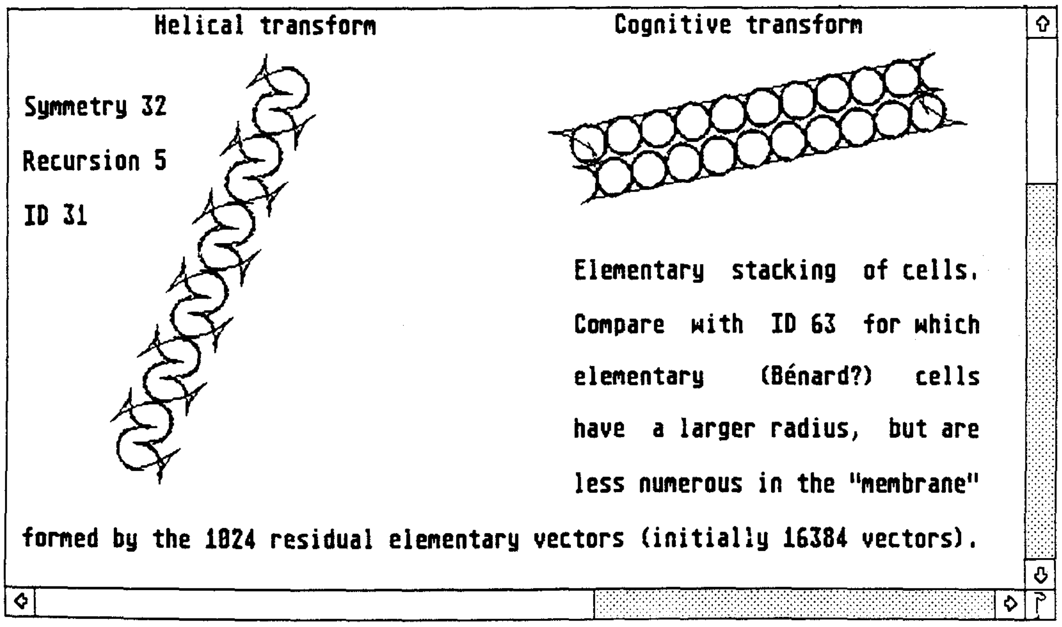


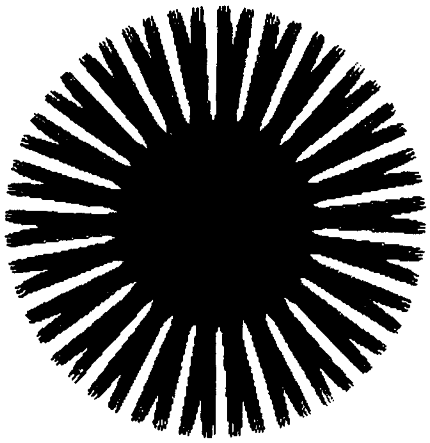
A flower from the APL garden : 89th ID of the cognitive transform for Recursion 5 in 15-fold symmetry.
The central crown exhibits 30-fold symmetry.
All the chestnut sub-shapes may be considered as "strange attractors" (see Chaos Theory).


An easily-written cover function allows to draw two shapes per screen :
Symmetry 13
Recursion 5
ID number 89 of the helical (left) and cognitive transforms (right).
The cursor sprite of the Atari (a bee) pays a visit to strange insects...


≠\ both preserves symmetry and creates chirality (helicity, vorticity) :
Symmetry 13
Recursion 5
ID number 127 of the helical (left) and cognitive transforms (right).
3 times 512 patterns for each symmetry are available with recursion 5.


Symmetry 16
Recursion 5
ID 127
THE CRYSTALLINE STRANGE ATTRACTORS
2 x 3
pattern of
octogons
2 x 5 1/2 pattern of
octogons, altogether
11 octogons.
How ≠\ a symmetry maker and a symmetry breaker at the same time, produces regular crystal-like patterns, introducing odd numbers spontaneously. Note. All genitons are cubic roots of the conforming binary unit matrices.


Helical transform
Cognitive transform
Symmetry 32
Recursion 5
ID 31
Elementary stacking of cells.
Compare with ID 63 for which
elementary (Bénard?) cells
have a larger radius, but are
less numerous in the "membrane"
formed by the 1024 residual elementary vectors (initially 16384 vectors).

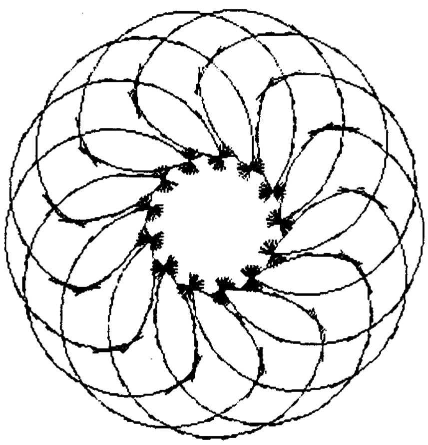


High symmetries may be explored; however, intermediate arrays can reach a large size.

Such a flower originally has 13,312 vectors, which reduce to 3,328, fortunately, already for Recursion 4.

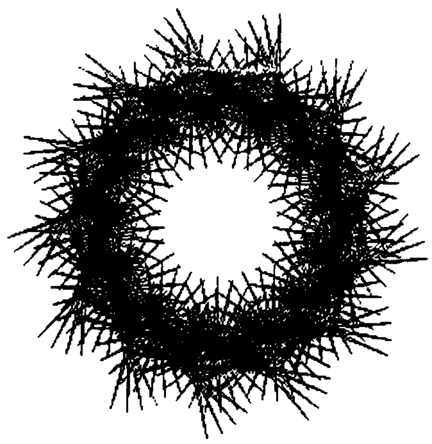
This is the equivalent of the von Koch's snowflake (which exhibits a 6-fold symmetry). Symmetry is 104.

Symmetry 104 Recursion 4 ID number 2 i.e. ≠\≠\ of



the Helical transform

(the birth of a torus ?)



the Cognitive transform