Utility model: (Software)

The use of zero (0) is excluded due to its nature.

This method consists of an array of nine (9) rules or numbers that interact with each other following a constant and unchanging pattern in a natural way. It is worth mentioning the possibility of altering said pattern to obtain altered results of said rules, but this is not the case for this model.

In a base ecosystem (9) it is possible to calculate, store, predict and / or move in both directions without altering the expected result. These properties are implemented in this model to compress and extract data from a fractal or numerical array.

Below is the logical order or arrangement of these numbers based on their properties:

Forwarding logic: [3,5,7,9,2,4,6,8,1]
Reverse logic: (main): [6,4,2,9,7,5,3,1,8]

It is observed that: their disposition varies in relation to the usual accepted and standardized order (1,2,3,4,5,6,7,8,9). Once we understand the above, we focus on the primary properties of said numbers or rules, the two most fundamental being the jump constant [K] and the parity or mirror of said numbers [KPN], this referring to the state of least entropy or genesis. This state is defined by the rules of (9) and from there all other rules (numbers) are derived. The respective couples formed by them are:

Forwarding pairs: [1,8] [2,7] [3,6] [4,5] [9,9]
Reverse pairs: [8,1] [7,2] 6,3] [5,4] [9,9]

(9)(n)(n1)
↓
(n2)

Kpn~(n) = (9 - n) = [K+-]

[K+] = $\overrightarrow{\text{Diff}}$ (Kpn~(n) + 1 ---> 9)

[K-] = (n)

$\overrightarrow{[ss]}$ ejemplo 1:

(9)32	(9)35	(9)37
122	125	127
32	35	37
5	8	1

ejemplo 2:

(9)72	(9)75	(9)77
162	165	167
72	75	77
9	12	14
	3	5

entonces:

if: (n2) < Kpn~(n): (n1) = (n2 - n) ---> [K-]
if: (n2) > Kpn~(n): (n1) = (n2 + Kpn~(n) ---> [K+]

See table (forwarding - Reverse)

Mathematical concept of a calculation neuron:

Since all the rules or numbers are related to its base (9) it is possible to extract all of them from one of them. With the exception of (3, 6) and the base itself. This is because these three values make up the base.

Example:

1 = (Kpn~1) = (8) → (8 – 1) = (7) → (Kpn~7) = (2) → (7 – 2) = (5) → (Kpn~5) = (4)

To obtain the above values 3,6,9 we can use:

Return to (9)
(8 – 5) = 3; (7 – 4) = 3; (5 – 2) = 3; (4 – 1) = 3
(8 – 2) = 6; (7 – 1) = 6

Advance to (9)
(1 + 2) = 3; (3 + 9) = (12 -→) = 3; (4 + 8) = (12→) = 3; (5 + 7) = (12 →) = 3; (6 + 6) = (12 →) = 3
(1 + 5) = 6; (2 + 4) = 6; (3 + 3) = 6; (6 + 9) = (15 →) = 6; (7 + 8) = (15 →) = 6

Advance to(9)
(1 + 8) = 9; (2 + 7) = 9; (3 + 6) = 9; (4 +5) = 9

Return to (9)
(1 – 1) = 9; (2 – 2) = 9; (3 – 3) = 9; (4 – 4) = 9; (5 – 5) = 9; (6 – 6) = 9; (7 – 7) = 9; (8 – 8) = 9;
(9 – 9) = 9

In this calculation method no values are added or subtracted, what is computed or calculated are the displacements and the direction of the same (advance, return) being possible to return in two different ways depending on which rule (number) initiates or formulates the request of displacement.

Example:

(8 – 5) = 3; (8 – (5 → 14) = 6

In the previous example we expand the value (5) to get 14, starting from this we subtract (8) to get (6). It is important to consider this point to define which rule makes the request for displacement since in reverse we can obtain two different valid values and linked to the calculation start factors.

Once understood the above we can arm our calculation neuron being this one (U4) (unsigned) of 4 elements or values (a, b, c, d). If we try to create a neuron of less than 3 elements we would obtain ambiguous results for the same calculation factors. In other words, we would have value collisions or more than one valid result.

U4:

Any neuron before starting the calculation process or integrating with the other components is considered U4, depending on the desired purpose or use it can be equalized to order the values or directly indicate the use or mode of calculation that is going to be used

In this example of data compression to obtain a fractal or mathematical summary of its components or initial factors we have used calculation mode R4 to load the information, LB for the processing and transmission (inheritance) of the same and SH to obtain the final result .

Hexa - C:

The neuron (Hexa -C) corresponds to a U6 model composed of an R4 and an inheritance connector which moves inside it to function as the data output element or inheritance to the next calculation neuron.

In other publications it will delve into the different types of neurons, as well as the use of them according to the direction of movement.

By understanding these concepts we can design equations or mathematical models for prediction, analysis, quantification and other purposes. The primary objective of this publication is the use of the above for data compression purposes (utility model: software) in a mathematical fractal that inherits in a binding manner the properties of all the elements that compose it. In this way we design an equation to decompose the final result of said calculation in the components or primary products as well as respect or maintain its correlation and / or location with respect to the other members of the initial calculation. This last property is fundamental at the time of reversing or decompressing a fractal, since it returns the data in the same way in which they were introduced. The foregoing being consistent and therefore (readable) for both a device or software as well as for humans.

In this (public) method some procedures are omitted for copyright (c) and commercial use reasons.

Julio C. Gómez.