

Characterisation of random series by the Delta-2 method.

Hubert Schaetzel

Abstract The purpose of this article is to describe a new method of characterizing the random or non-random nature of a strictly increasing sequence of real numbers. This discovery is a mere coincidence. Not being at all a specialist in probabilities and statistics, the aim here will not be to provide a theoretical justification for the process : we start from a definition and verify the agreement with this definition on standard examples. The proposed test is based on the Delta-2 algorithm.

Caractérisation de suites aléatoires par la méthode du Delta-2

Résumé L'objet de cet article est de donner la description d'une méthode nouvelle de caractérisation de la nature aléatoire, ou non, d'une suite de nombres réels. Cette découverte est un simple fruit du hasard. N'étant pas du tout spécialiste de probabilités et de statistiques, le but ne sera pas ici de fournir une justification théorique du procédé : nous partons d'une définition et vérifions la concordance à cette définition sur des exemples types. Le test proposé repose sur l'algorithme Delta-2.

Statute Preprint
Date Version 1 : July 10 of 2020

Summary

1. Preamble.	2
2. Test implementation.	2
3. Discussion.	3
3.1. The integers.	3
3.2. The prime numbers.	4
3.3. The polynomial function.	5
3.4. Pseudo-random number generators.	7
3.5. The logarithmic and exponential functions.	7
3.6. The trigonometric functions.	9
3.7. The zeroes of the Riemann Zeta function.	12
4. Disadvantage of the method.	13
5. Appendix 1.	14

1. Preamble.

The statistic of random samples obeys different laws of probability, Gauss-Laplace normal law, Bernoulli law, binomial law, Pascal law, Student law, hypergeometric law, Zipf law, Poisson law, Pareto law, continuous uniform law, Fischer-Snedecor law, χ^2 law, ..., where this list can be extended almost to infinity [1].

Many procedures ensure that a particular sample obeys such a law, tests of normality, Jarque-Bera test, Kuiper test, Shapiro-Wilk test, Anderson-Darling test, Cramer-Von Mises test, χ^2 test... These methods of comparison are nevertheless less numerous than the previous ones. The spectral test on the other hand allows evaluating the quality of a generator of pseudo-random numbers of linear congruential type.

The test proposed here is closer to this second category. We are interested here in whether a list derived from a given function is random or not.

Our tool being the Delta-2 algorithm, we define this character by adding "randomness under Delta-2 testing".

Delta-2 is a process of accelerating the convergence of series in numerical analysis attributed to Seki Kōwa, Japanese mathematician (late 17th century) and popularized by the mathematician Alexander Aitken in 1926 [2]. This algorithm is distinguished by its simplicity and efficiency (see & Test implementation.)

This algorithm is not used here for the usual purpose of determining the value to which the studied sequence converges, but to recognize the randomness or not of a series of numbers.

These numbers may be positive or negative numbers in any order, **but** must be sorted according to a monotonous (strictly) growing series before testing.

The study below focuses on the random nature of a sequence by a binary response : yes or no. We will see the disadvantages of the method to get for a finer ranking.

2. Test implementation.

Let us have (x_1, x_2, \dots, x_n) a series of strictly positive numbers whose random or not nature we wish to characterize.

Let us have :

$$T_1(x_n) = x_n - (\Delta x_n)^2 / \Delta^2 x_n \quad (1)$$

where

$$\Delta x_n = x_{n+1} - x_n \text{ et } \Delta^2 x_n = x_{n+2} - 2x_{n+1} + x_n \quad (2)$$

This writes also

$$T_1(x_n) = (x_{n+2} \cdot x_n - x_{n+1}^2) / (x_{n+2} - 2x_{n+1} + x_n) \quad (3)$$

This algorithm is applied in the Delta-2 method recursively. Successive results are given in a table that follows :

1	x_1	$T_1(x_1)$	$T_2(x_1)$...	$T_m(x_1)$
...
n-4	x_{n-4}	$T_1(x_{n-4})$	$T_2(x_{n-4})$...	
n-3	x_{n-3}	$T_1(x_{n-3})$	$T_2(x_{n-3})$		
n-2	x_{n-2}	$T_1(x_{n-2})$	$T_2(x_{n-2})$		
n-1	x_{n-1}	$T_1(x_{n-1})$			
n	x_n	$T_1(x_n)$			
n+1	x_{n+1}				
n+2	x_{n+2}				

Usually, for a converging sequence (x_1, x_2, \dots, x_n) , there is an acceleration of convergence of the series $(T(x_1), T_2(x_1), \dots, T_m(x_1))$. It is even possible by this method to find converging sequences derived from divergent series.

The exploitation of the table is done here by comparing the series (x_1, x_2, \dots, x_n) to the series $(T(x_1), T_2(x_1), \dots, T_m(x_1))$.

1	x_1	$T_1(x_1)$
2	x_2	$T_2(x_1)$
...
m	x_m	$T_m(x_1)$
...	...	
$n = 2m+1$	x_n	

As the table shows, the algorithm provides m values for the $n = 2m+1$ initial values. The cardinal of the initial series is double of the final sequence. The comparison is therefore made on the first half of the initial series.

The series $(x_1, x_2, \dots, x_i, \dots, x_m)$ and $(T(x_1), T_2(x_1), \dots, T_i(x_1), \dots, T_m(x_1))$ define two trajectories written as $Traj(i, x_i)$ and $Traj(i, T_i(x_1))$. Subsequently, we will simplify the writing of a trajectory $Traj(i, w_i)$ by simply writing it $Traj(w_i)$.

Definition 1.

The list or series (y_i) is random under Delta-2 test if only if the $Traj(x_i)$ and $Traj(T_i(x_1))$ trajectories are nearly adjacent for most of the definition domain, (x_i) being the ordered increasing series of (y_i) .

Remark.

The author understands the vagueness of the notion of neighbourhood here. The reader will refer to the examples to remove this ambiguity.

In these examples, we represented the initial $Traj(x_i)$ trajectory in red and the final trajectory $Traj(T_i(x_1))$ in blue. The abscissa axis is coordinated by i.

3. Discussion.

3.1. The integers.

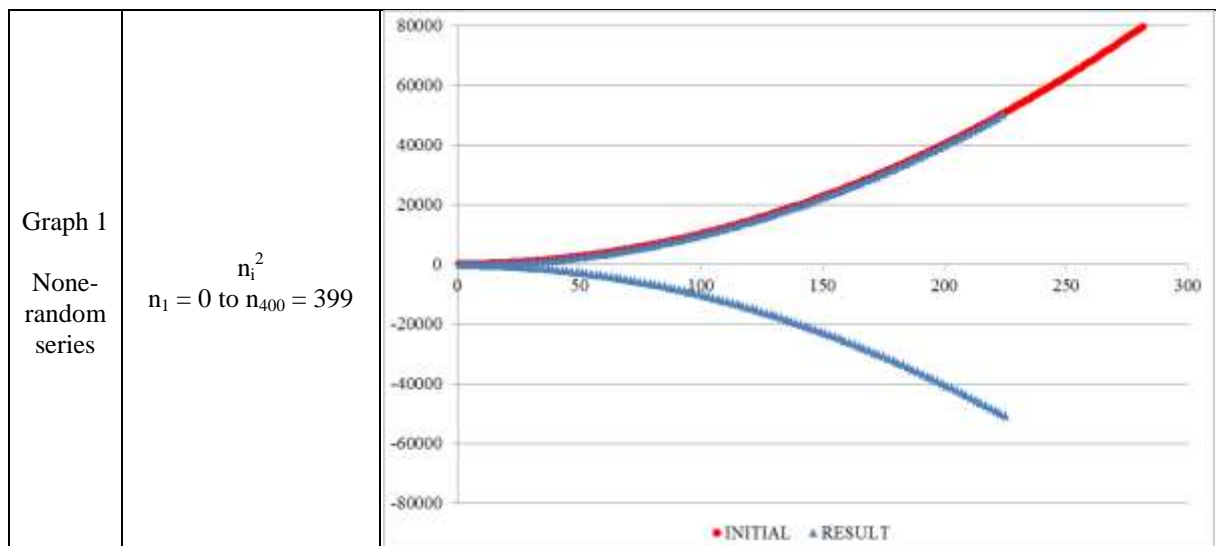
Let us have $(0, 1, 2, 3, \dots)$, the sequence of integers.

Clearly this sequence is not a random series. If we submit it to the Delta-2 algorithm, we get $T_1(x_n) = (x_{n+2} \cdot x_n - x_{n+1}^2) / (x_{n+2} - 2x_{n+1} + x_n) = ((n+2) \cdot n - (n+1)^2) / (n+2 - 2(n+1) + n) = -1/0$. Thus the trajectory of the result of the algorithm does not at all then overlap with the initial trajectory, since not defined.

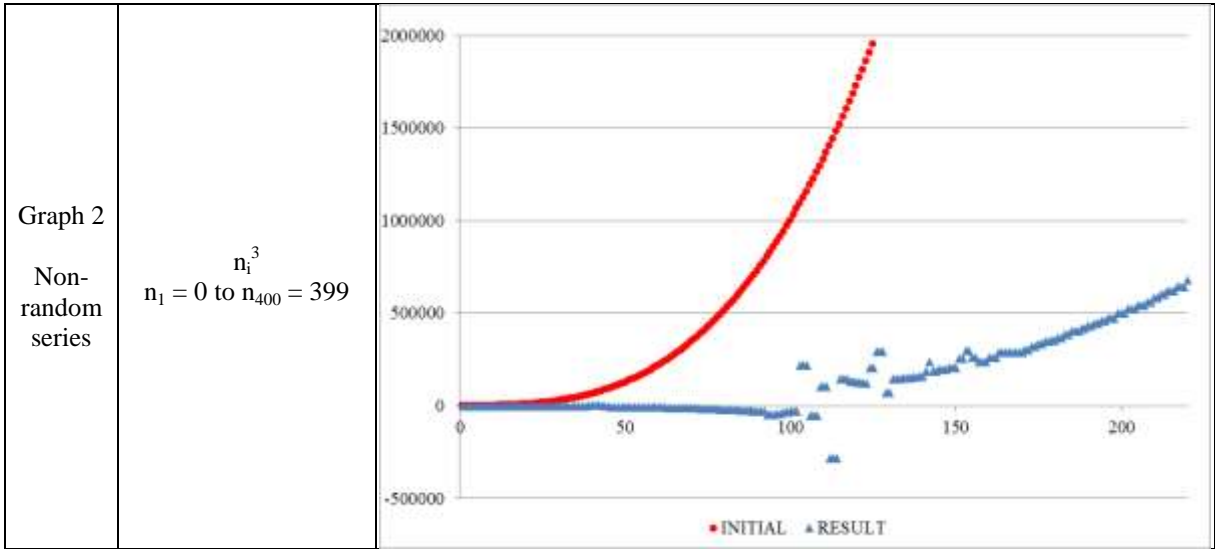
Hence the series of the integers is indeed a non-random sequence under Delta-2 testing.

However, our answer is a little premature precisely because the image trajectory is not defined. So let us rather consider $(0^2, 1^2, 2^2, 3^2, \dots)$ the series of squares of the integers. If the series of integers is not random, this second sequences should not be neither. Graph 1 gives a comparison of the initial and final trajectories in this case. The final trajectory consists of two branches, one overlapping and the other being symmetrical to the abscissa axis.

However, according to our definition for the classification of a list (definition 1), the final data must in general nearly overlap the initial trajectory in order to identify it with a random sequence, which is clearly not observed here half of the time.

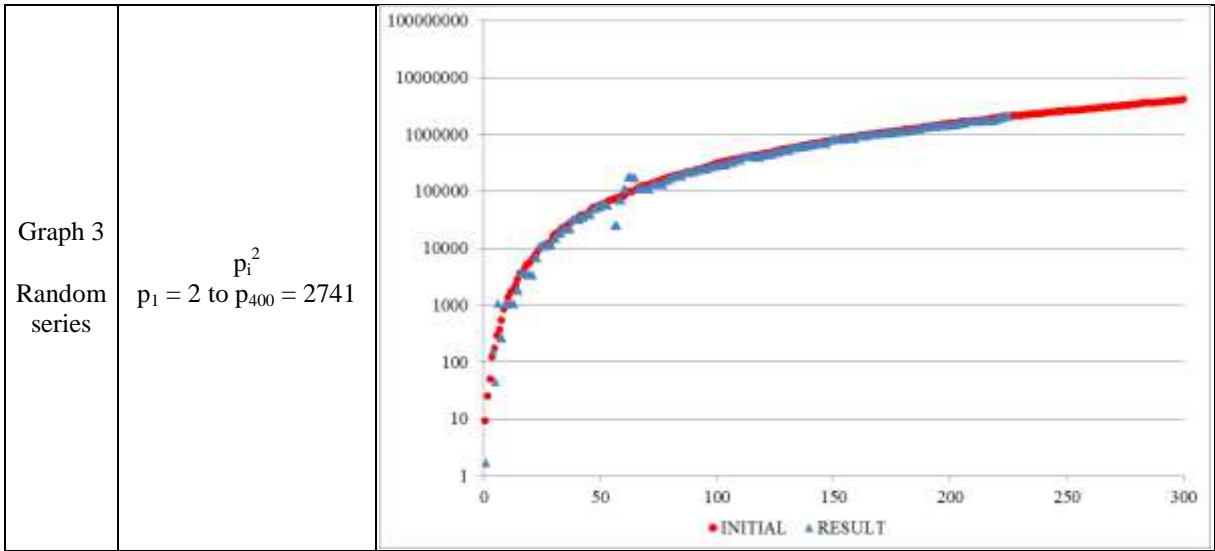


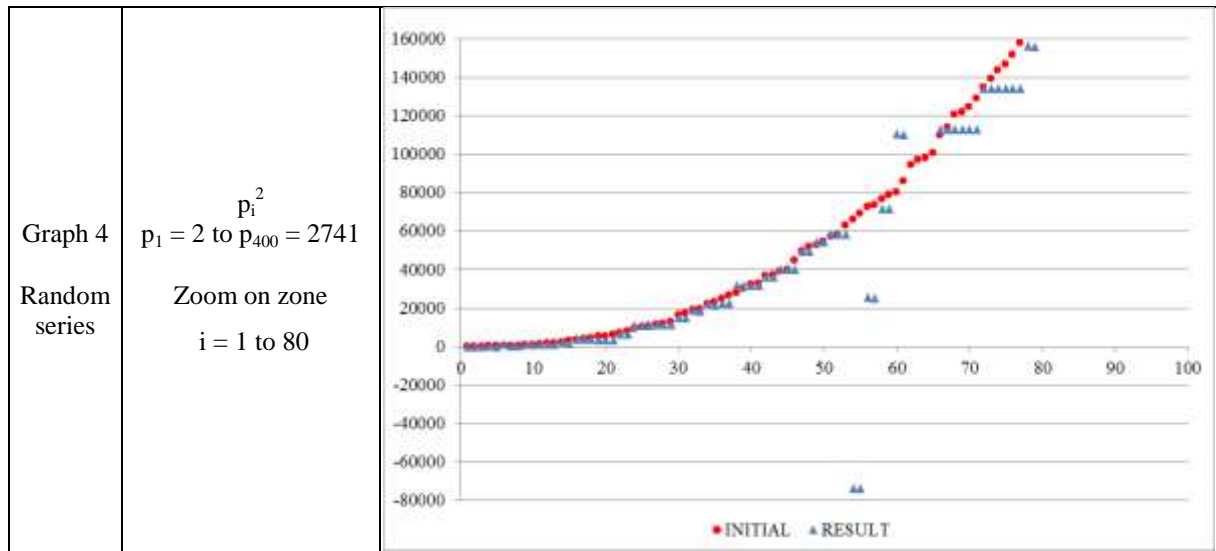
For the fussy reader, we move on to the series $(0^3, 1^3, 2^3, 3^3, \dots)$ where the ambiguity is definitively discarded (see graph 2).



3.2. The prime numbers.

Let us have now (p_1, p_2, \dots, p_n) the series of n^{th} first numbers prime numbers.
Is this sequence random ?
If we submit it to the Delta-2 algorithm, we get $T_1(p_i) = (p_{i+2} \cdot p_i - p_{i+1}^2) / (p_{i+2} - 2p_{i+1} + p_i)$. Here, the outcome $p_{i+2} - 2p_{i+1} + p_i = 0$ occurs for a very large number of p_i values. (for examples 3, 47, 151, 167, 199, 251, 257, 367, 557, 587, 601, ...). We are facing with the same problem of ambiguity as with the series of natural integers. We treat this point in the same way as before by squaring terms. The initial and final trajectories are given by graph 3. The ambiguity is removed in the same way as before and the conclusion this time is that the series of the prime numbers is random under Delta-2 testing.

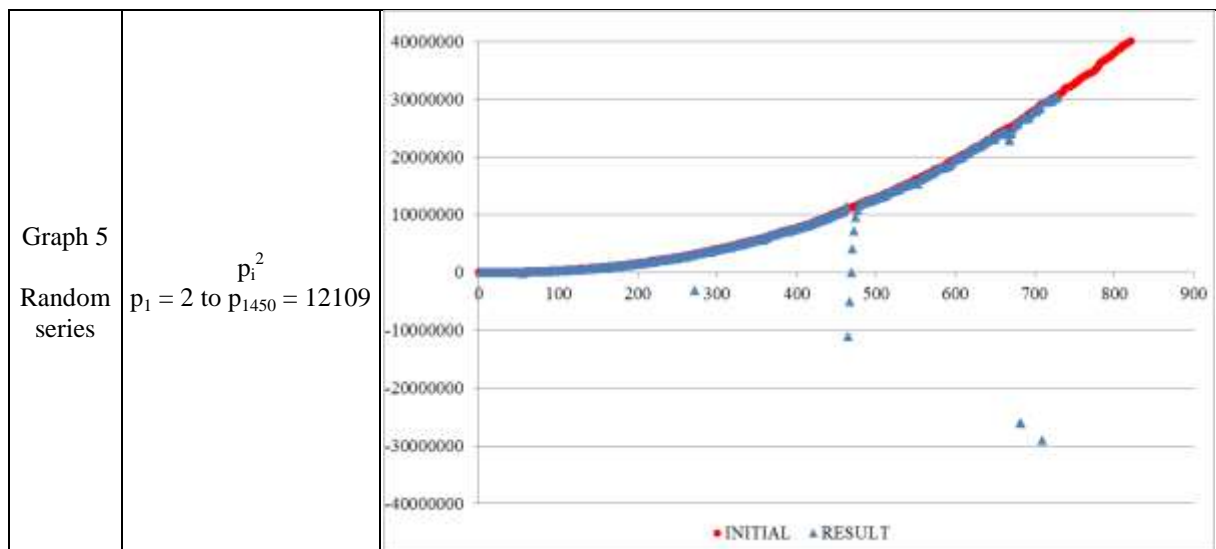




The reader will have noticed, however, an area of disturbances at $i = 54$ ($p_{54} = 251$, $p_{55} = 257$, $p_{56} = 263$, $p_{57} = 269$). The same phenomenon is repeated, for example, at $i = 271$ ($p_{271} = 1741$, $p_{272} = 1747$, $p_{273} = 1753$, $p_{274} = 1759$), $i = 464$ ($p_{464} = 3301$, $p_{465} = 3307$, $p_{466} = 3313$, $p_{467} = 3317$), $i = 682$ ($p_{682} = 5101$, $p_{683} = 5107$, $p_{684} = 5113$, $p_{685} = 5119$) and $i = 709$ ($p_{709} = 5381$, $p_{710} = 5387$, $p_{711} = 5393$, $p_{712} = 5399$).

This is the outcome of $p_{i+2} - 2p_{i+1} + p_i = 0$ repeated at i and $i+1$. The origin is the presence of constellations (p , $p+6$, $p+12$, $p+18$). Constellations of this type are also present at $i = 3$ (5, 11, 17, 23), $i = 5$ (11, 17, 23, 29), $i = 13$ (41, 47, 53, 59) and $i = 18$ (61, 67, 73, 79), but in these cases there are one or more additional prime numbers within each constellation (that is (7, 13, 19), (13, 19), 43 and 71) which avoids the phenomenon.

Recovery to the vicinity of the initial trajectory is more or less rapid depending on the case.



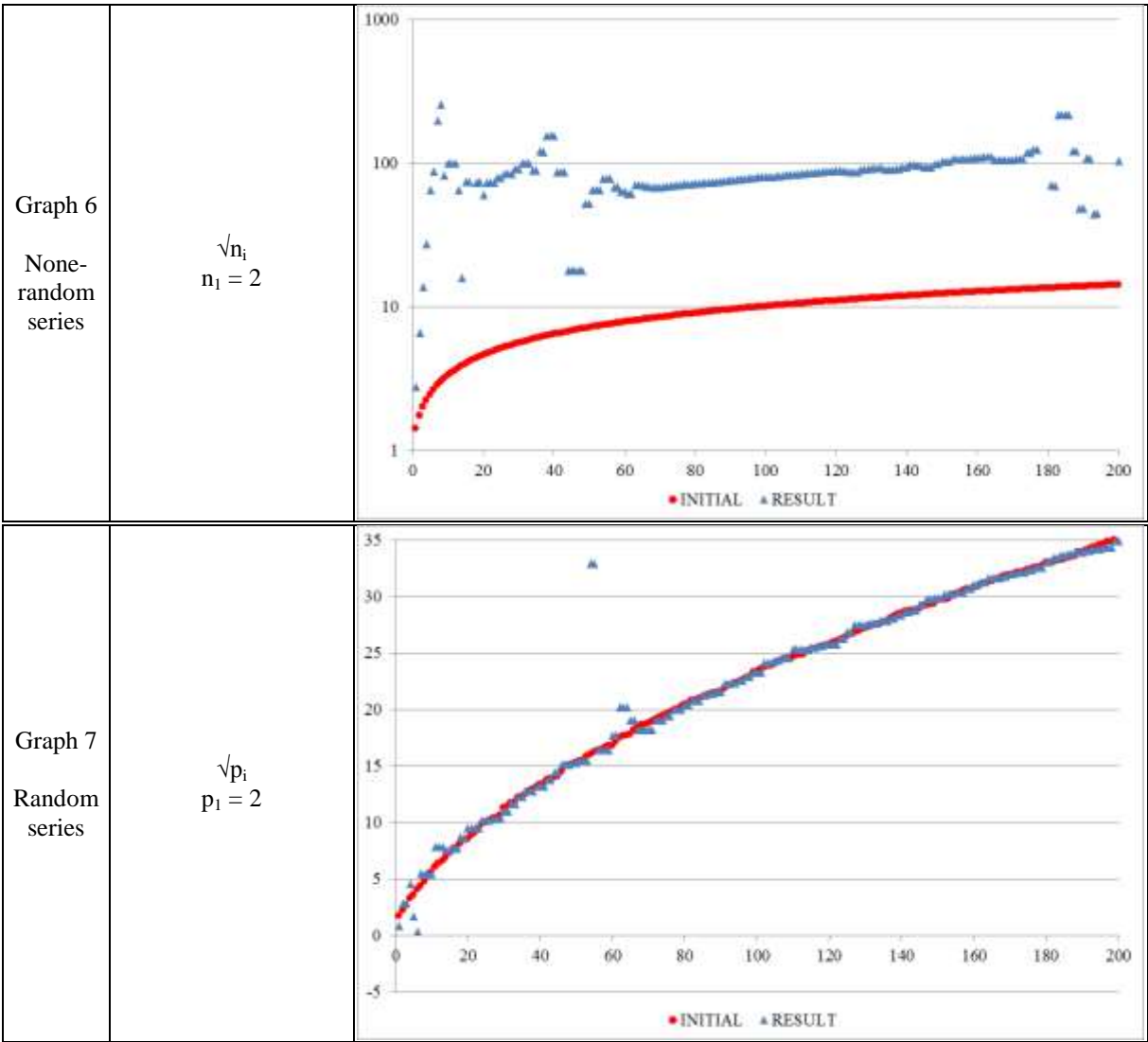
3.3. The polynomial function.

Monomials, integer-degrees polynomials and real-degree polynomials transmit the nature of the series of integers (see graphs 6 and 8) and prime numbers (see graphs 7 and 9).

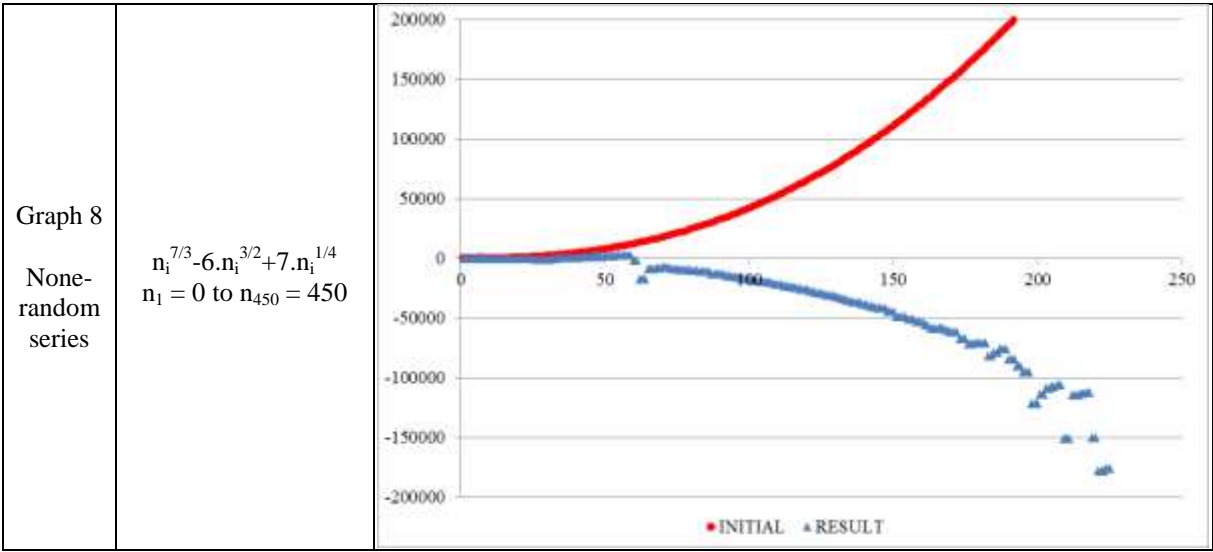
This seems a general result in the sense that initially taking in a function f to be tested n_i or $P(n_i)$ is equivalent (same conclusion for $f(n_i)$ and $f(P(n_i))$) and similarly for p_i or $P(p_i)$, the only difference is sometimes the ambiguity resulting from the use $f(n_i)$ which does not appear with $f(P(n_i))$.

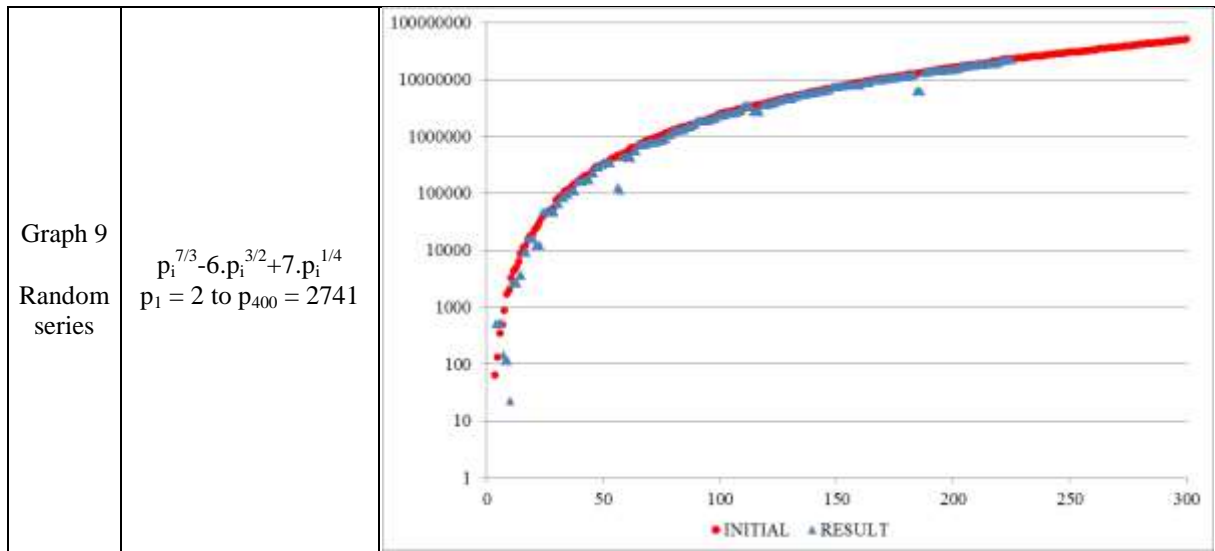
The "polynomial" function therefore has a significant practical aspect.

Example 1



Example 2



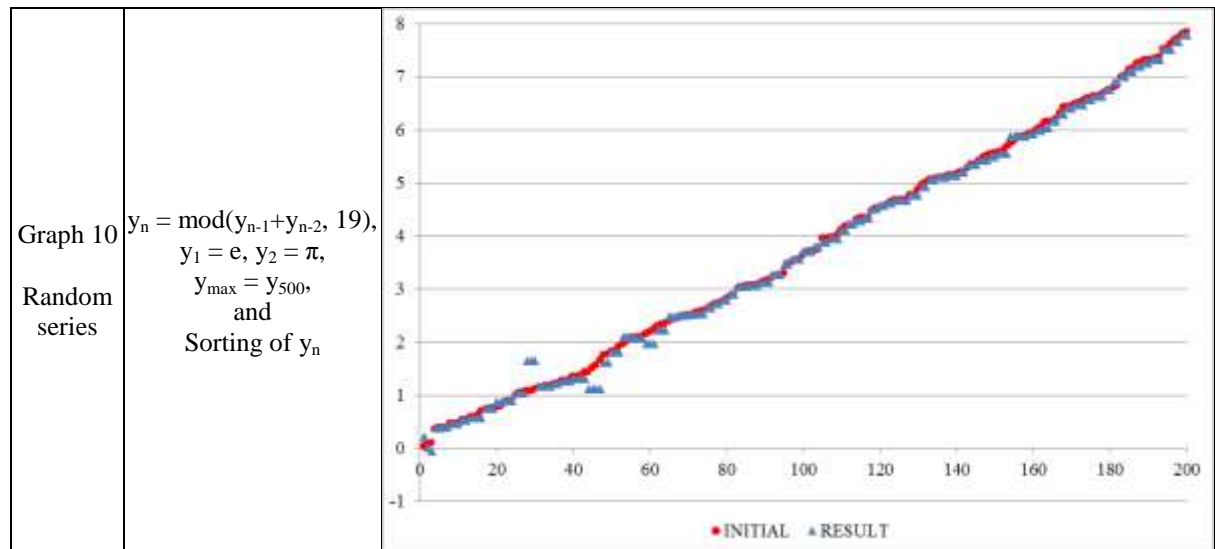


3.4. Pseudo-random number generators.

Here we chose an example based on a Fibonacci series. Number y_n is obtained by adding y_{n-1} and y_{n-2} modulo 19, with the initial y_1 and y_2 values being close to $\exp(1)$ and π . We produce from there, the series y_i up to y_{500} . The list includes $y_3 = 5,859874482$, $y_4 = 9,001467136$, $y_5 = 14,86134162$, $y_6 = 4,862808753$, $y_7 = 0,724150371$, and so on. The list is then sorted by increasing numbers $x_1 = 0,004659205$, $x_2 = 0,039129397$, $x_3 = 0,068068362$, $x_4 = 0,103515556$, $x_5 = 0,350805214$, $x_6 = 0,381762519$, ..., $x_{500} = 18,93342327$.

The test is applied to the series (x_i) in order to evaluate the (y_i) .

The trajectories overlap approximately, which means that the sequence is random under Delta-2 testing.

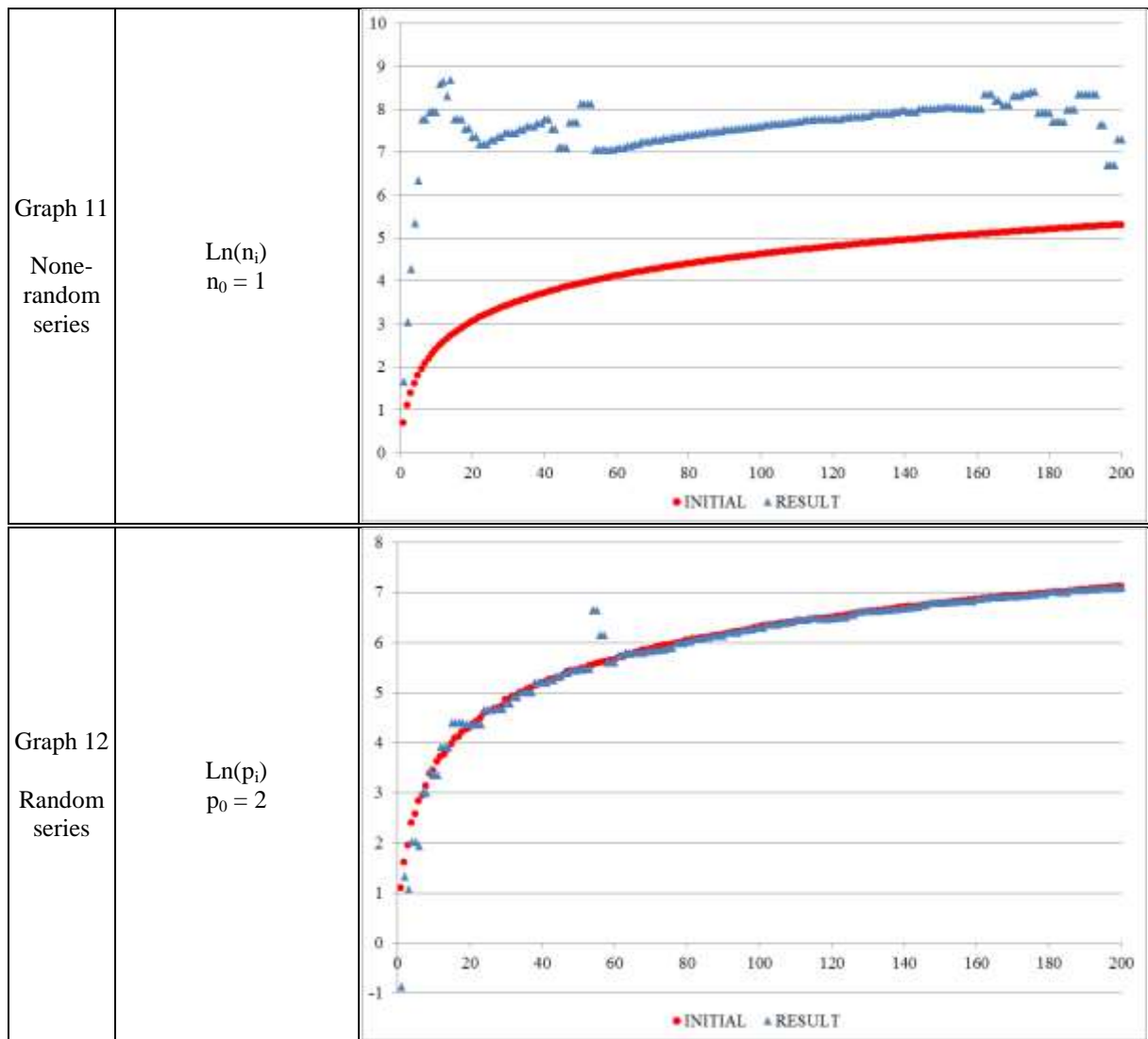


Again, we observe local "abnormal" fluctuations. The origin of the phenomenon is very close to the reason given above, namely this time it comes either from the incidence $x_{n+2} - 2x_{n+1} + x_n \approx 0$, or more generally from $T_i(x_{n+2}) - 2T_i(x_{n+1}) + T_i(x_n) \approx 0$, since the effect can manifest itself at any stage of the evaluation.

3.5. The logarithmic and exponential functions.

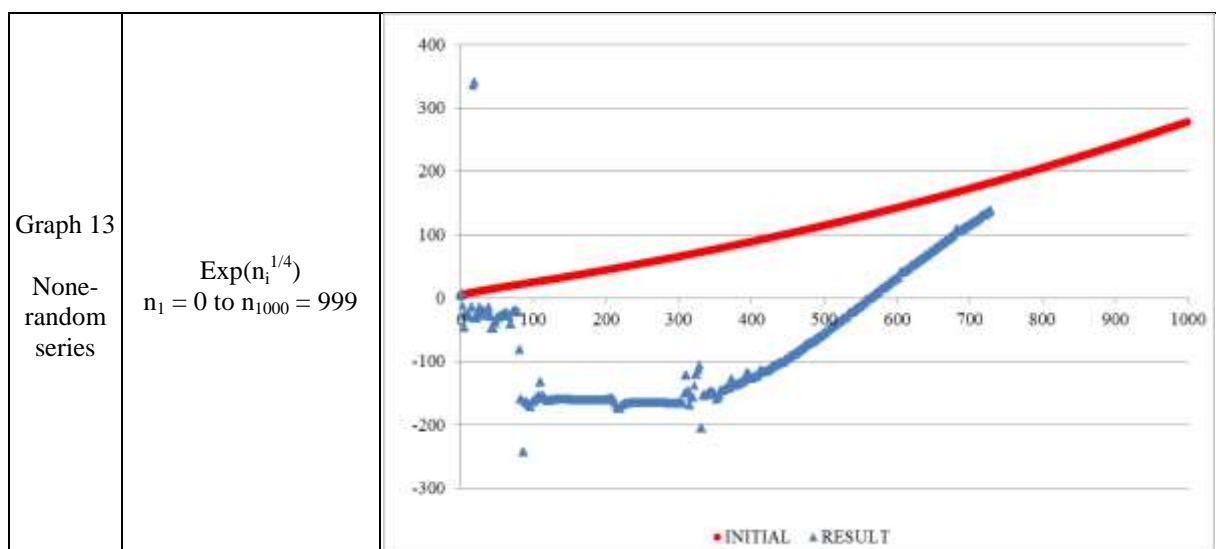
The observation of graph 11 shows that $\ln(n)$, $n = 1, 2, 3, 4, \dots$ is not a random sequence under Delta-2 testing. The function has the same behaviour as monomials, integer or real degrees polynomials have on the series of integers (see graph 8).

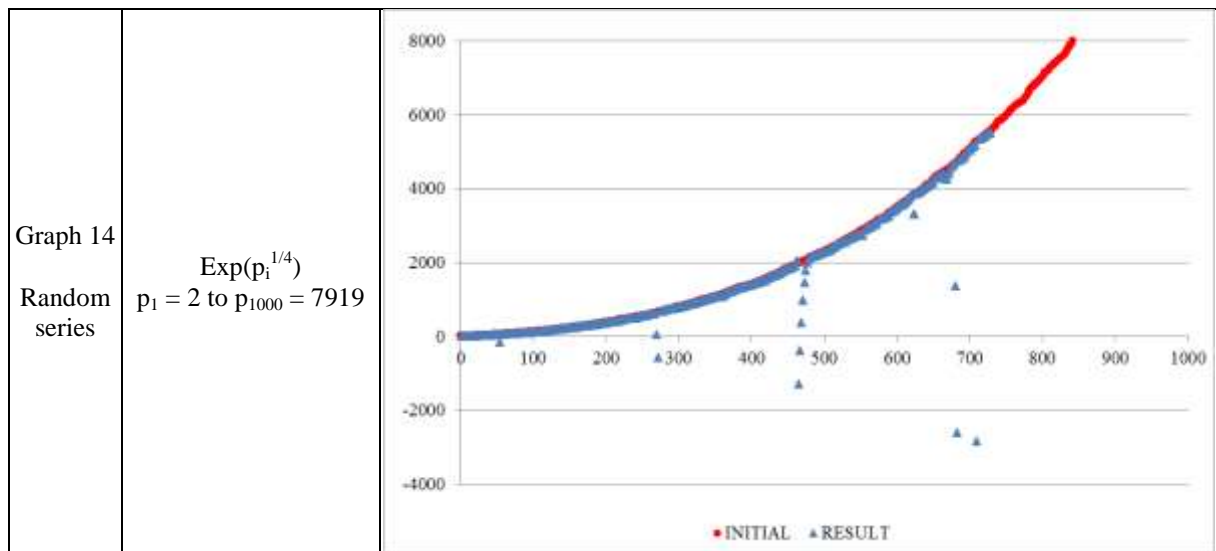
We expect in this case that this function will keep the random series property applied to the prime numbers, which is indeed the case. (see graph 12).



Note that the effect of constellations (such as $p_{54} = 251$, $p_{55} = 257$, $p_{56} = 263$, $p_{57} = 269$ at $i = 54$) remains quite obvious here. Thus, it crosses the exponential function, reciprocal function of the logarithm, in the same way as we show underneath.

The application to the exponential function of n_i^a with $a = 1$ (degree 1) raises the same disadvantages and disturbances as those already expressed. This is solved by taking a $\neq 1$ (here $a = 1/4$).

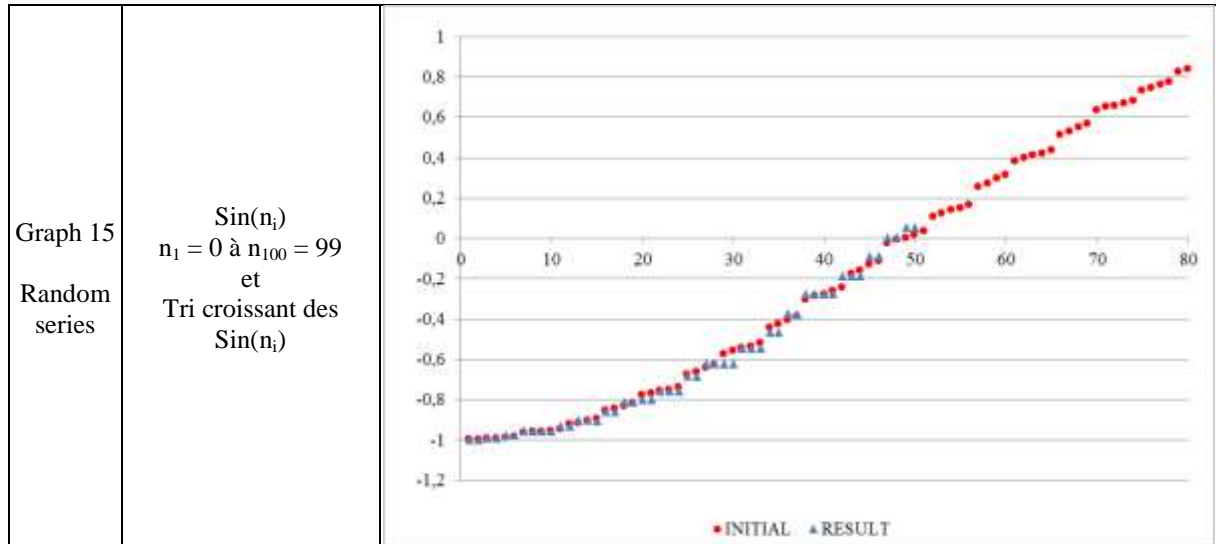


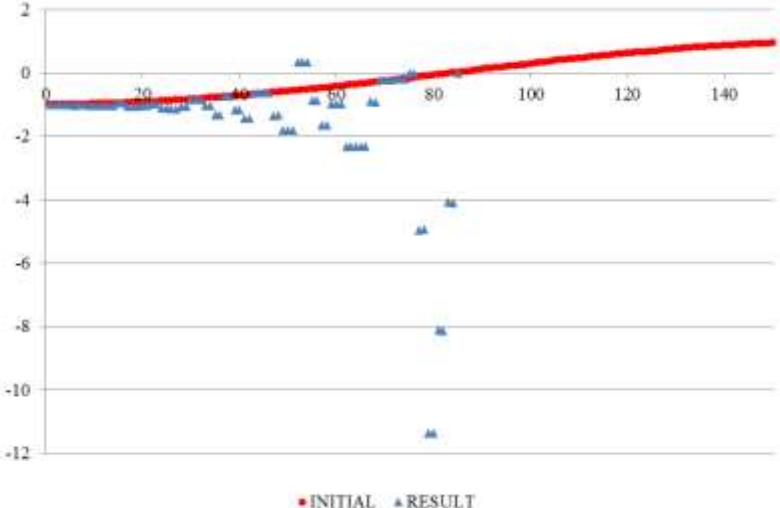
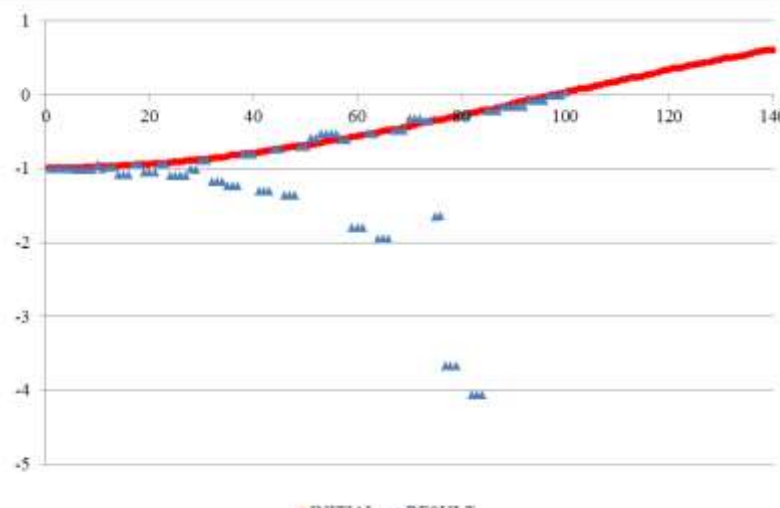
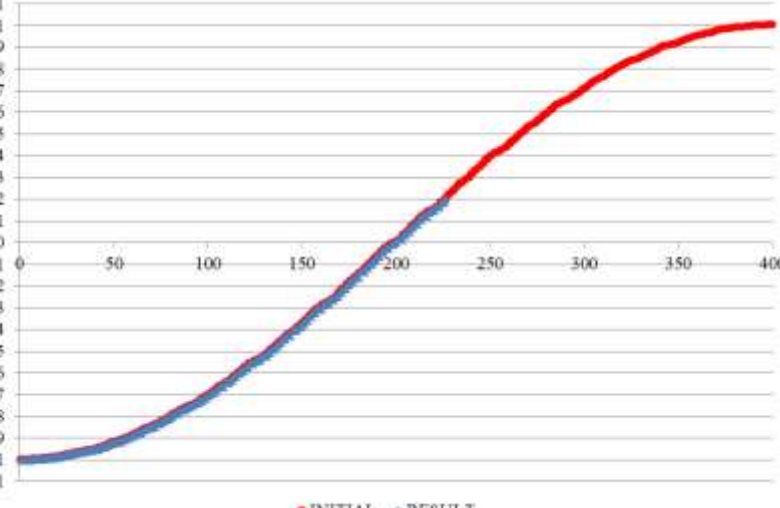


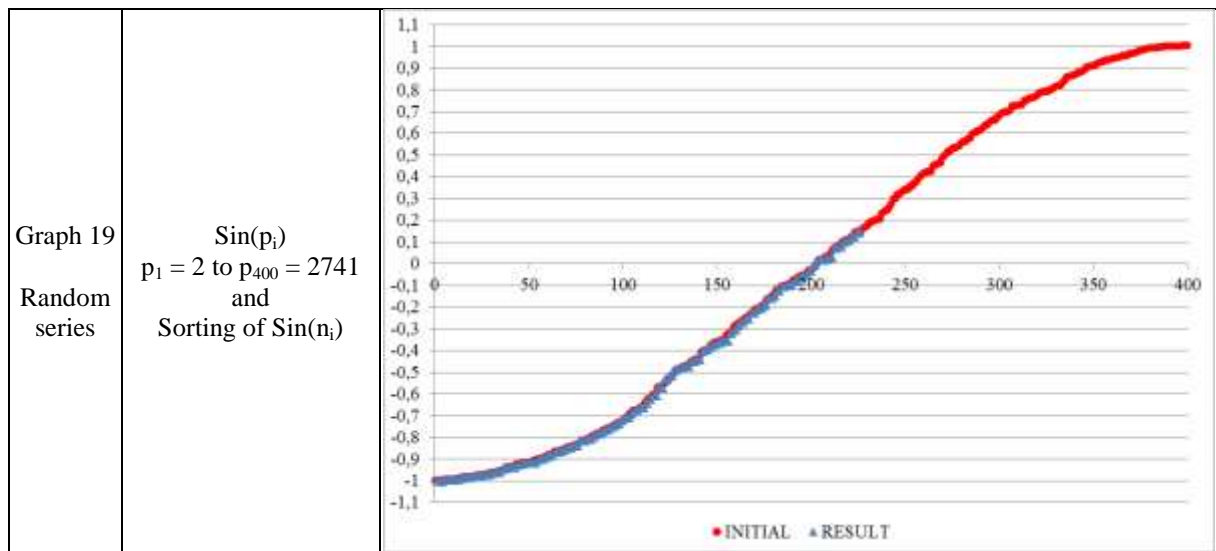
3.6. The trigonometric functions.

The choice of using a periodic function to obtain random numbers is not, a priori, a very wise decision. If this approach is retained, however, it is obvious that some choices have trivial non-random results. For example, the list $\sin(n.\pi)$, with n integers, is not systematically zero. Similarly, $\sin(n.\pi/100)$ becomes a very bad choice for a number of increments greater than 100. Even when the value of π is taken very coarsely, the incidence of $x_{n+2} - 2x_{n+1} + x_n \approx 0$ soon manifests itself, creating large ambiguities here and there.

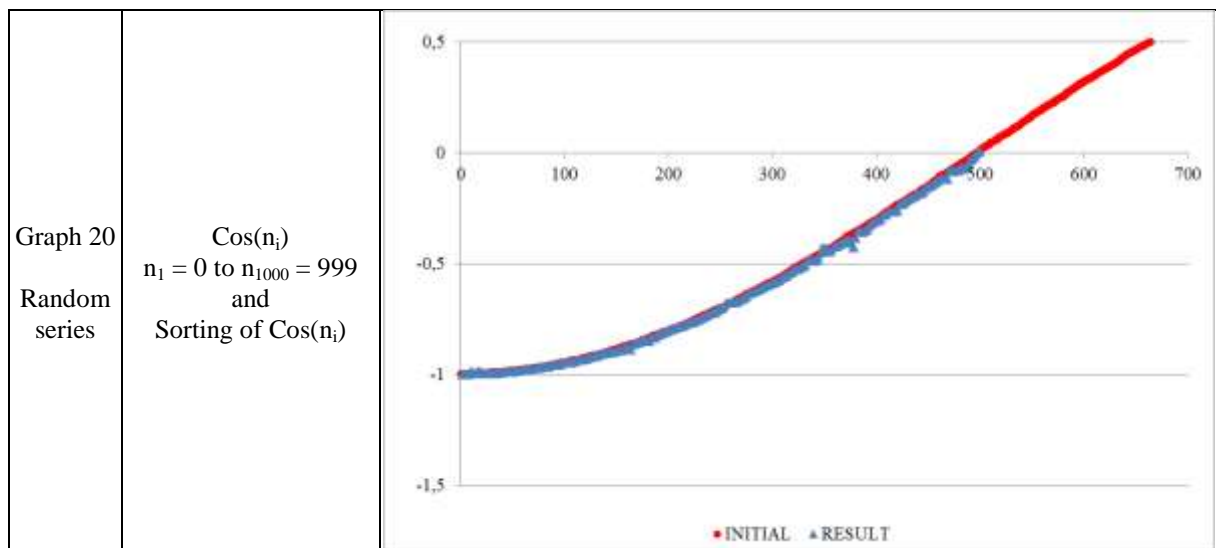
Below are some results for the Sine function. By applying it to integers, we get an initial random series behaviour (graph 15), then the list becomes non-random (graphs 16 and 17), then becomes random again (graph 18). With the choice of prime numbers, the "random series" message is much more consistent (general aspect of graph 19 with more or less deviations or anomalies).



<p>Graph 16</p> <p>None-random series</p>	<p>$\sin(n_i)$ $n_1 = 0$ to $n_{170} = 169$ and Sorting of $\sin(n_i)$</p>	
<p>Graph 17</p> <p>None-random series</p>	<p>$\sin(n_i)$ $n_1 = 0$ to $n_{200} = 199$ and Sorting of $\sin(n_i)$</p>	
<p>Graph 18</p> <p>Random series</p>	<p>$\sin(n_i)$ $n_1 = 0$ to $n_{400} = 399$ and Sorting of $\sin(n_i)$</p>	



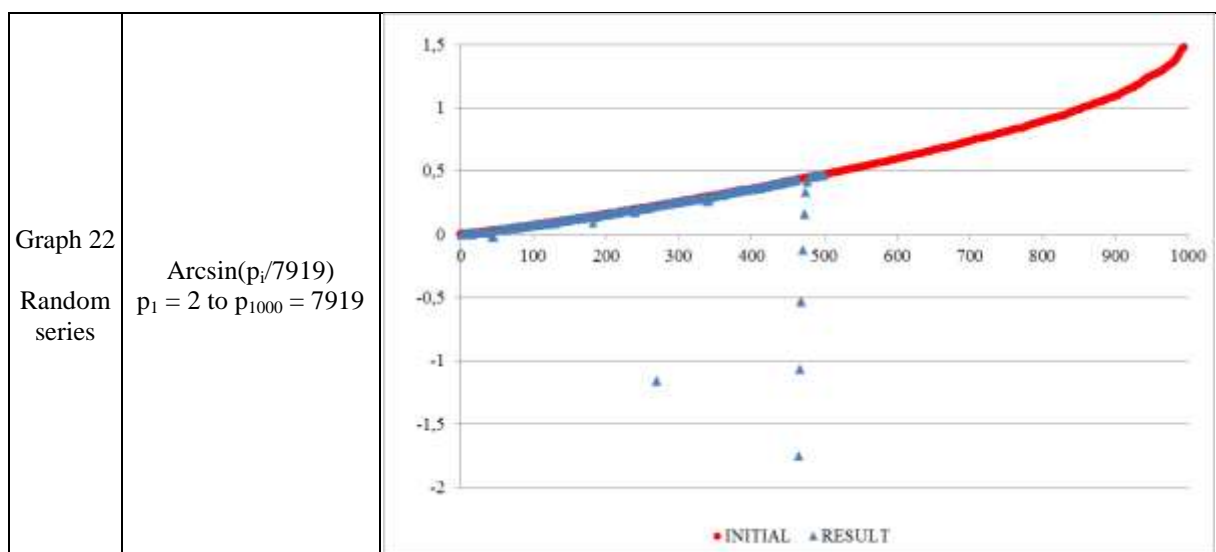
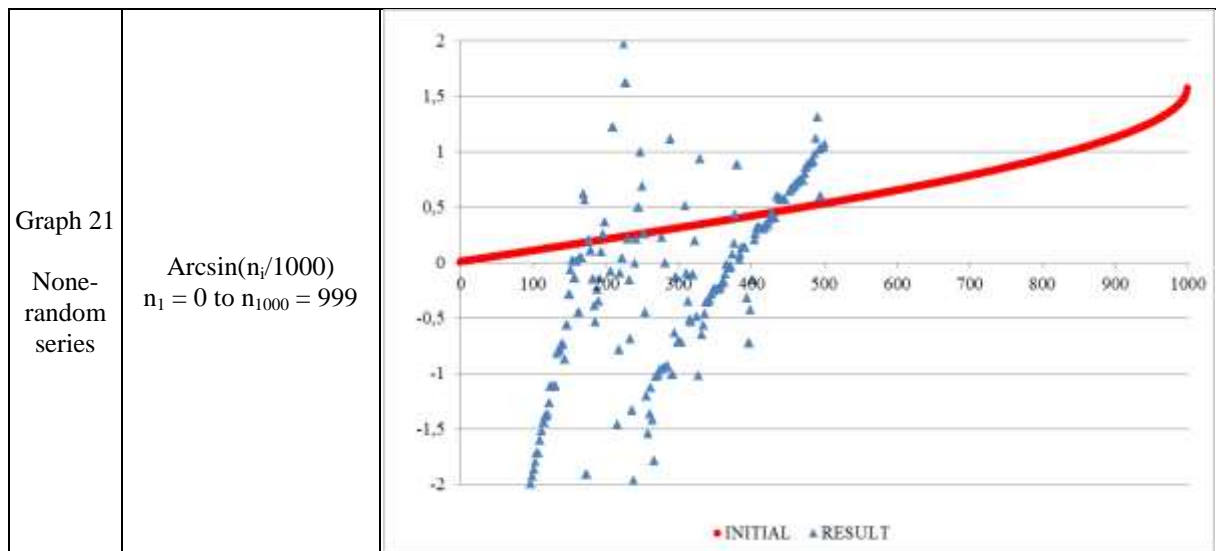
The $r < n_i < s$ areas where the $\text{Cosine}(n_i)$ function behaves like a random series are very different from the Sine function areas. For example, the series is grossly non-random up to $s = 500$, then random in the range $r = 500$ to $s = 1000$, then becomes non-random again (at least up to $s = 1500$).



The Tangent function applied to integers combine the Sine and Cosine shortcomings.

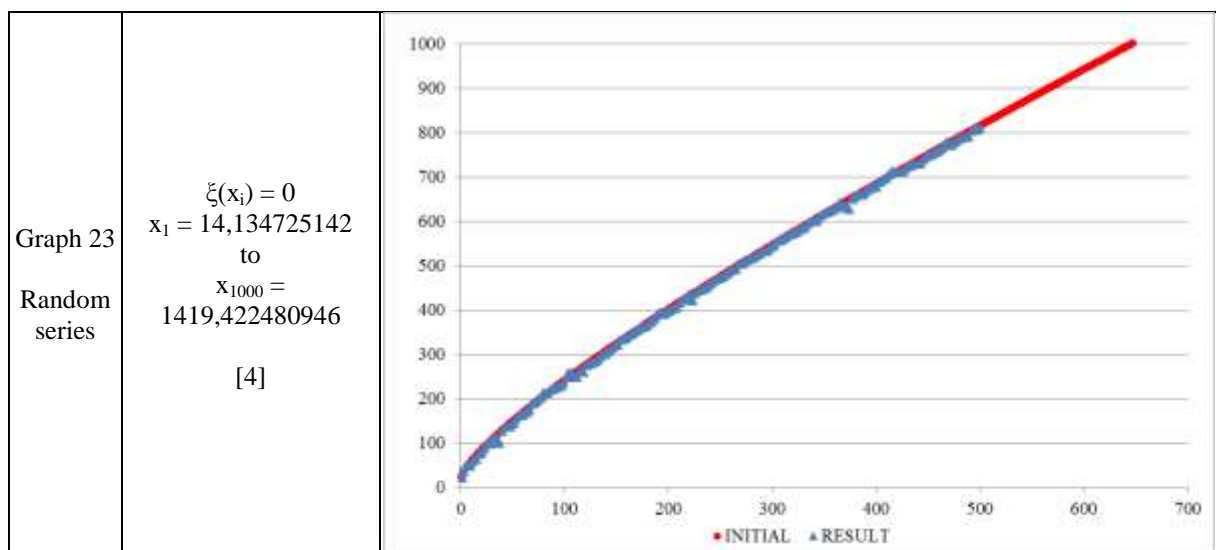
This discussion is therefore a warning against the value of using these functions as a generator of pseudo-random numbers.

The reciprocal functions of periodic trigonometric functions (Arcsine, Arccosine, Inverse Tangent) behave unambiguously, reproducing the state of the series to which they are applied. Assuming that the nature of reciprocal functions is the same as that of the initial functions, since it is in fact only a permutation of the x and y axis, we obtain a definitive answer to the actual nature of the Sine, Cosine and Tangent functions.



3.7. The zeroes of the Riemann Zeta function.

The Riemann Zeta function zeroes are indeed random numbers under Delta-2 testing.



4. Disadvantage of the method.

The main drawback described repeatedly in the examples, namely the repeated outcome $x_{n+2}-2x_{n+1}+x_n \approx 0$, is not really a drawback to answering the question " yes or no the list is random? ". But, in order to compare the degree of random behaviour between two lists, using for example the values obtained by the standard deviation of the two trajectories of the first list and then of the second list, these outcomes are indeed very detrimental. Should they be taken into account or excluded from the calculation ? The question is open.

REFERENCES

- [1] https://fr.wikipedia.org/wiki/Liste_de_lois_de_probabilit%C3%A9.
- [2] https://fr.wikipedia.org/wiki/Test_de_Kolmogorov-Smirnov.
https://en.wikipedia.org/wiki/Wald-Wolfowitz_runs_test.
https://fr.wikipedia.org/wiki/Test_du_%C2%B5.
https://fr.wikipedia.org/wiki/Test_spectral.
- [3] <https://fr.wikipedia.org/wiki/Delta-2>.
https://fr.wikipedia.org/wiki/Seki_Kowa.
https://fr.wikipedia.org/wiki/Transformation_de_Shanks.
- [4] List of zeroes of Zeta function.
http://www.dtc.umn.edu/~odlyzko/zeta_tables/index.html
- [5] <http://sites.google.com/site/schaetzelhuberdiophantien/>
- [6] <https://hubertschaetzel.wixsite.com/website>

5. Appendix 1.



Examples of programming with PARI : :

Example 1 :

Fibonacci series : $a_n = \text{Modulo}(a_{n-1} + a_{n-2}, 19)$, $a_1 = e$, $a_2 = \pi$

```
{default(realprecision, 30); \\ to choose
nb = 500; \\ to choose
nb = 2*(nb/2\1);
x = vector(nb,i,0); y = vector(nb/2,i,0); a = vector(nb,i,0); b = vector(nb,i,0);
x[1] = exp(1); x[2] = Pi;
for(n = 3, nb, x[n] = (x[n-1]+x[n-2])%19);
a = vecsort(x);
print("Initial list");
for(i = 1, nb, print(a[i]));
for(c = 1, nb/2-1,
for(r = 1, nb-2*c,
den = a[r+2]-2*a[r+1]+a[r];
num = a[r+2]*a[r]-a[r+1]*a[r+1];
if(den == 0, b[r] = 0.0, b[r] = num/den));
y[c+1] = b[1];
a = b);
print("Final list");
for(i = 1, nb/2, print(y[i]))}
```

Example 2 :

Sine function applied to a prime numbers list.

```
{nb = 1000; \\ to choose
nb = 2*(nb/2\1);
x = vector(nb,i,0); y = vector(nb/2,i,0); a = vector(nb,i,0); b = vector(nb,i,0);
for(n = 1, nb, x[n] = sin(primes(nb)[n]));
a = vecsort(x);
print("Initial list");
for(i = 1, nb, print(a[i]));
for(c = 1, nb/2-1,
for(r = 1, nb-2*c,
den = a[r+2]-2*a[r+1]+a[r];
num = a[r+2]*a[r]-a[r+1]*a[r+1];
if(den == 0, b[r] = 0, b[r] = num/den));
y[c+1] = b[1];
a = b);
print("Final list");
for(i = 1, nb/2, print(y[i]))}
```