Decryption of Evolutionary Fractals by means of Procedural Randomness generated with an Intelligence Model based on the CSN Algorithm

Catalin Silviu Nutu[1]

[1] Constanta Maritime University, 104 Bd. Mircea cel Batran, Constanta, Romania
nutu_catalin@yahoo.com

Abstract. The idea that the intelligence of Nature may be embedded in fractals has already been stated by author, since 2016, as in [2] and [4]. This paper is expanding on previous papers of the author [1] and [3], by providing an alternative intelligence model to complete the present model of A.I. This paper also introduces two new concepts: the concept of evolutionary fractal and the concept of procedural randomness. The intelligence model presented in this paper is based on the CSN Algorithm. This CSN Algorithm can be used, on one hand, to decrypt evolutionary fractals having unknown rules, and on the other, it can be used to simulate the way in which random creative processes occur. The CSN Algorithm simulates better how brain behaves, because when making scientific discoveries, often times, existing information is not processed as in classical A.I models, which are mostly deterministic, but it is processed aleatory by brain, in random processes, as in the model based on the CSN Algorithm. Decryption of these evolutionary fractals may be the key of understanding of human brain functioning and of human intelligence. The first step, however, is the decryption of evolutionary fractals related to more simple lifeforms, and only then, after the decryption at this first step, the decryption of the human intelligence may be addressed. The model presented in this paper can either be used as an entirely independent intelligence model to simulate human intelligence random creative processes, or it can be used to create enhanced intelligence models by joining the model presented in this paper, based on the CSN Algorithm together with the present classical model of A.I. In a further section of paper, other additional possible applications of this algorithm are presented.

Keywords: A.I., evolutionary fractal, CSN Matrix, CSN Algorithm, randomness, trial and error, disruptive process, procedural randomness

1 Introduction

The A.I., in its present stage, as presented in the literature in [5], [6], [7], [9] and [10], is hardly to be called „intelligence" since it is not possessing at least the intelligence of primitive life forms of Nature. In this context, by Nature it is to understand all non-living or living structures.

Excepting some basic mechanisms, for example, like it is the perceptron's model, primitively simulating the functioning of brain cells, the stage of understanding human brain and human intelligence, has remained pretty much the same, unchained since the ancient times by the first Greek thinkers like Socrates, Plato and Aristoteles, as presented in [11].

More than 2 millennia ago, they were the first ones to think about human logic and intelligence, and also the first ones to seriously think about thinking. The main thing added by logic and science thereafter was mainly technology related inventions, without any spectacular advances in thinking since then.

The idea that only a small amount of reality is available to our limited human senses and the rest of it remains elusive, is in agreement with the ancient thoughts of Plato in The Republic VII, where people in the cave were only to perceive shadows of reality.

In order to detect invisible structures, inaccessible to human senses and human logic and science, indirect methods, not logic based, but based on methods like the one presented in this paper, can be employed to find those structures remaining elusive to human detection and to human logic.

Another reason why present day A.I. cannot be called intelligence is that Nature is still not intelligible enough for man, in order that he himself is able to create something, at least at the level of these structures of Nature.

The idea of fractal like shape of information knowledge and intelligence was already presented by author of this paper, back in 2016, in [2].

One of the main concepts to describe and understand Nature is the concept of fractal. Almost any form of living or non-living structure is based on this concept of fractal, hence his paramount importance for the understanding of Nature. The first step toward an artificial real intelligence would be that Nature is made intelligible enough from the viewpoint of its rules, by means of understanding its fractals.

If Someone (God) or Something (Nature) has triggered evolutionary process of life forms, of human brain and of the human intelligence, then the secrets of a performant A.I. lie in decoding the fractals associated with the respective evolutionary processes.

Therefore, until understanding evolutionary fractal structures, both of them, intelligence in general and human intelligence in particular, will remain an unsolved riddle and a mystery.

Fractal structures are expression, evidence and proof of intelligence of Nature, which is however, far more advanced than any human creation. Understanding and decrypting fractals may be the first significant step toward understanding and imitating intelligence of Nature, since the intelligence of Nature, as it has already also been stated in [1], may be embedded in the rules of evolutionary fractals.

This paper expands on and refines many of the ideas presented in previous papers of the author [1], [2] and [3], namely that the intelligence of Nature may be actually embedded in fractals and unknown fractal rules, that human brain intelligence processes may also evolve according to not yet known fractal processes and that often times intelligence processes are governed by randomness, accidents, chance or trial and error.

This paper, however, takes these ideas to another next level by introducing new ideas and concepts such as the concepts of evolutionary fractal and procedural randomness. It also introduces a new type of algorithm, the CSN Algorithm, which is based not on only one CSN Matrix but on two different CSN Matrices. A Python script corresponding to the presented CSN Algorithm is presented as well.

While the computer MATLAB script presented in the paper [1] was merely only a very simple example with explanatory purposes, which explained the concepts and

theory introduced in the respective paper, the Python script presented in this paper has a much more general form.

Another idea related to the main tool presented in this paper, which is namely the CSN Algorithm, is that it can also be used as an encryption instrument, in a manner which will be explained later on.

The CSN Algorithm in this paper is based on the two CSN Matrices, the CSN Matrix of elements and the CSN Matrix of operations.

The elements and operations in the cells of these two matrices are aleatory chosen for the Python script mirroring the CSN Algorithm presented in this paper.

In reality, depending on the area in which it is employed, the elements and operations can be specific elements and operations, depending on the respective field in which the CSN Algorithm is employed. This is to say, in order to apply this algorithm in a certain field, one has to have previous knowledge about the respective field and about the elements and operations governing that specific field.

Depending on a certain input value, the CSN Algorithm outputs a certain result. The results obtained by using this algorithm are ranging from very small-scale domain values up to large scale domain values. The CSN Algorithm is thoroughly explained throughout this paper, and it represents the main tool to generate procedural randomness, procedural randomness which is in turn used to decrypt evolutionary fractals.

2 Method

2.1 The research question: Decryption of Evolutionary fractals by means of procedural randomness in order to understand Nature

Evolutionary fractals may be deemed as structures in which is embedded the Intelligence of Nature and the more exact definition of them is presented, together with the definition of the term procedural randomness, later in this paper.

Besides as being defined as fractals structures evolving from previous states into subsequent other different ones, the evolutionary fractals are characterized by sometimes, a sudden disruptive change of the fractal rule.

Evolutionary fractals are defined as fractals evolving from a certain shape into another one, by changing shapes because of moving through different fractal rules, or in a particular case, from a life form into another, from an initial state of the life form into a final state of life, by passing through a multitude of phases and transformations. More exactly it means a structure based on and constructed using a set of many fractals, each one having its own fractal rule.

In order to decrypt highly complicated human brain intelligence processes or to understand disruptive evolutions of Life and of Nature, one cannot disregard the paramount importance or randomness or of trial and error processes. The first step toward decrypting the Intelligence of Nature could be exactly decryption of the evolutionary fractals. Understanding highly complicated human brain intelligence processes may be a next step in understanding intelligence processes, after previously decrypting simpler evolutionary fractals encountered in Nature.

This CSN Algorithm proposed in this paper has mainly the same objectives as in [1] and [3], namely, on one hand, as a general encryption/decryption tool, to be used to encrypt/decrypt coded messages or to decrypt this special type of evolutionary fractals.

The CSN Algorithm can be used as encryption tool in the following way. Both, the random value encryption key Tkey and the output value of the CSN Algorithm itself, should be firstly transmitted over a secure transmission line, such as for example a quantum key distribution secure system.

The further secure communication between the entities involved in communication (e.g. Bob and Alice) will be ensured by checking the correspondence between the Tkey and the output, using two additional encryption levels: the first of them is the CSN Matrix of Elements and the second one is the CSN Matrix of Operations. If the Tkey and output of algorithm correspond by applying these both matrices which are integral part of the algorithm, then the encrypted communication is thus additionally ensured.

On the other hand, it can be used to generate intelligence models which are more closely related to the actual intelligence processes taking place in the brain, when new breakthrough discoveries are made in scientific research.

These superior intelligence models, taking into account the randomness of the intelligence processes will be called from now on enhanced intelligence models (EI models).

In order to work with and to decipher particular processes or phenomena, intended to be decrypted using the CSN Algorithm presented in this paper, may, however, involve and require previous specific data and previous specific knowledge about the data cells of the both CSN Matrices involved: the CSN Matrix of elements and the CSN Matrix of operations.

2.2 Evolutionary fractals and procedural randomness. The need for randomness in highly complicated/creative intelligence processes

The new concepts introduced are the concepts of evolutionary fractal and of procedural randomness.

The concept of evolutionary fractal is to be understood in the way it is presented summarily above and it refers to the generalization of the fractal concept. Evolutionary fractals are defined as fractals structures evolving from previous states into subsequent other different ones, concomitantly with, sometimes, a sudden disruptive change of the fractal rule.

The concept of evolutionary fractal introduced by the author of this paper is defined as being related, but somehow in opposition to the known concept of evolutionary algorithm, because of one significant difference.

Evolutionary algorithms stand for small successive changes and improvements, mostly implying fitting and adapting in order to survive, such as in the case of the Darwinian law of evolution.

In the case of humans, for example, the above law of evolution is somehow reversed, as presented in [12]. Big problems and significant issues of the mankind are not solved by well adapted herds of people, but by those weird, exceptional not adapted brilliant people.

Moreover, great breakthrough discoveries in science and technology are not smooth processes, but they are highly disruptive processes, which may be simulated by procedural randomness which can be generated by using the CSN Algorithm presented in this paper.

Hence, disruptive transitions (such as the transition from huge reptiles/dinosaurs to mammals, or highly disruptive thinking processes such as new ideas and theories in science, cannot be modeled or simulated by evolutionary algorithms, but by evolutionary fractals which are able to generate procedural randomness by means of the CSN Algorithm.

Evolutionary fractals are in contrast with the evolutionary algorithms, randomly generated disruptive evolutions in Nature, not necessarily implying fitting and adapting, though, implying evolution.

The term of procedural randomness means that a random input value generates a certain output value depending on the interval to which the random input value pertains, and it is procedural because it is generated by means of an algorithm, namely the CSN Algorithm. In other words, a certain random input value activates a certain cell element and a certain cell operation in each of the CSN Matrices associated to the CSN Algorithm, thus generating procedural randomness.

2.3 Objective and data used

Based on the ideas presented above, this paper has as its main objective to present a model of a special type of E.I., based on the proposed CSN Algorithm, which is used in both: to decode evolutionary fractals, but also as an independent intelligence model.

The actual data used is represented by the functions and operations which correspond to the cells of the CSN Matrices, on which the CSN Algorithm is based upon. The algorithm presented in this paper is mainly based on the two different CSN Matrices: The CSN Matrix of Elements and the CSN Matrix of Operations.

The CSN Algorithm presented in this paper and its corresponding Python script do not refer to a specific field. The elements and the operations in the cells of the two matrices are randomly chosen, out of explanatory purposes and for exemplification of the application of the algorithm.

2.4 The necessity for an algorithm generating procedural randomness in order to understand intelligence processes in Nature

The intelligence model presented in this paper generates procedural randomness in order to decrypt evolutionary fractals and it is exclusively based on the CSN Matrices and on the CSN Algorithm theory introduced by the author.

A new kind of E.I., however, partly procedural and partly randomly can also be deemed and employed to decrypt complicate evolutionary fractal rules, which then are to be used to simulate the real intelligence of Nature.

So, the CSN Algorithm in this paper can either be used as the foundation of an entirely independent intelligence model to simulate human intelligence creative processes, or it can be used to create an enhanced intelligence model by joining the model

presented in this paper, based on the CSN Algorithm, together with the present classical model of A.I.

This new kind of A.I., partly procedural and partly random based could be employed to decrypt complicate evolutionary fractal rules, which then are to be used to simulate the real intelligence of Nature. Only after decoding simple life evolutionary fractals by means of procedural randomness generated with this algorithm, the simulation or the imitation of human brain will be possible, based on these identified and discovered evolutionary fractal rules.

In the previous papers [1] and [3], the CSN Matrix presented had combined elements and operations mixed up with another. The element of novelty in this paper is this new type of algorithm which combines the CSN Matrix of Elements with the CSN Matrix of Operations and links them both in this new type of algorithm named the CSN Algorithm.

In the existent A.I. models, once probabilities are assigned to certain processes, the respective processes become almost purely deterministic ones. The functioning of the human brain, however, is based on a combination between deterministic and quantum models, whereas present day A.I. is only based on deterministic and stochastic models which hardly, only somehow, try to imitate neural processes in human brain.

Moreover, actual functioning of human brain it may be based, in reality, on a combination of different fractal processes, which is an older idea which links the fractals with the quantum physics.

There is a significant chance that some of these fractal processes taking place in Nature and also in the human brain, as part of Nature, could be deciphered by testing a very large number of possibilities or possible solutions.

At its present stage of development, A.I. could in the present be employed, in the first instance, to identify such unknown fractal rules, and only then A.I. may subsequently be developed according to those rules governing the functioning of Nature and implicitly of the human brain.

It could be that, only by deciphering unknown fractal rules of the human brain, its actual functioning might be really revealed, known and understood.

The algorithm presented in this paper, may be simply only one of the tools we possess to discover the real deep intelligence of Nature we are part of.

In order to be closer to the denomination of „intelligence" A.I. should function much more similarly to the way human brain is functioning. The human brain is able to identify, even based on his very reduced information capacity and relatively slow computing speed, a variety of many different sets of valid solutions and methods to a certain problem or a set of problems.

Decoding of such evolutionary fractals, many of them encountered in Nature around us, may be one of the first steps in decoding the basics of Nature. The shapes, the processes and phenomena in Nature are mostly not deterministic, nor logic, neither iterative or partially stochastic process, as in the present A.I., but they are pretty much the opposite of all these.

Breakthrough discoveries in science or technology are neither logical nor deterministic processes, but they are, on the contrary, highly disruptive processes such as presented in [12].

Hence, a decryption based on and taking into account the importance of random and chance, such as many processes in Nature or such as in many human solutions and inventions, may be one of the best approaches of the Intelligence of Nature.

This is why, because of this randomness or because of unknown rules governing intelligence processes, scientists realized that quantum processing could play a crucial role in intelligence processes of human brain.

2.5 How procedural randomness is generated in order to decrypt evolutionary fractals: The CSN Algorithm and its outputs

The CSN Algorithm presented in this paper is one of the possible solutions to simulate procedural randomness which could eventually lead to decryption of evolutionary fractals in Nature.

This is also why the input data of the algorithm used, which in the CSN Algorithm has actually the denomination of Tkey, will be randomly generated.

The random input value Tkey determines the positions of the element and the position of the operation which are selected to generate the output of the CSN Algorithm

The output of the algorithm will be thus generated using random input data, and hence it will be able to generate procedural randomness, according to the explanations in the sections above.

Rather than generating new data based on both, huge amounts of data and huge computing power, like in the methods used by A.I., the proposed model seeks to find hidden rules and truths in the respective data, based on the CSN Algorithm presented and on its related necessary computing power.

The CSN Algorithm proposed and presented, could not only help to unveil some secrets of Nature, but it could also have the potential to decrypt intelligent designs in Nature.

The CSN Algorithm in short

1. Choose Tkey as a random number, Tkey =randint(0, m*n-1) depending on which the CSN Algorithm calculates the output
2. Generate the CSN Matrix of Elements and the CSN Matrix of Operations
3. The random input value Tkey determines the positions of the element and the position of the operation which are selected for the output of the CSN Algorithm. Identify the function element in the CSN Matrix of Elements and the operation element in the CSN Matrix of Operations corresponding to the random input value of Tkey
4. Apply the CSN Algorithm to the random input value Tkey using the corresponding element and the corresponding operation in the CSN Matrices

Out of simplicity and clarity reasons the CSN Matrices of Elements and Operations in the particular algorithm presented and in its corresponding computer script, are chosen with the same number of lines and columns.

The numbers of lines and columns, however, can be increased depending on the requirements of application of the algorithm.

The steps of the particular CSN Algorithm used for the Python script

As already mentioned above, when applied to specific fields, the CSN Algorithm may have specific function elements and specific operation elements, corresponding to the respective field.

The CSN Algorithm and its corresponding Python script in this paper, however, are presented only for explanatory purposes, since they do not address a specific area, the respective functions are not functions derived out of the prior knowledge of the respective field, but they are randomly generated.

The steps of the particular CSN Algorithm and of its corresponding Python script, used for explanatory purposes are as follows:

I. Choose the number of lines and columns of the CSN Matrices

II. Randomly generate the input value of the CSN Algorithm (Tkey)

III. Generate the CSN Matrix of Elements as defined in the Python script, as follows:

1) a lines (a =randint(1, m//3) of (f) polynomial functions
2) b lines (b=randint(1, m//3) of (g) polynomial functions
3) c lines (c= randint(1, m//3) of (h) random arguments of logarithmic operations
4) d lines (d=m-a-b-c) of (i) random arguments of exponential operations

IV. Generate the CSN Matrix of Operations as defined in the Python script, as follows:

1) a lines (i =randint(1, m//3) of operations of Differentiation
2) b lines (j =randint(1, m//3) of operations of Integration
3) c lines (k =randint(1, m//3) of operations of logarithmic transformation
4) d lines (d=m-a-b-c) of operations of exponential transformation

V. Depending on the Tkey value, the element chosen from the CSN Matrix of Elements and the operation chosen from the CSN Matrix of Operations are determined

VI. Applying the algorithm to the value of Tkey, which is random input argument for the operation and for the element used, produces the output of the algorithm

3 Results and further possible applications and uses of the CSN Algorithm

Depending on the random input value Tkey for the CSN Algorithm, the numerical outputs of the CSN Algorithm can range from very small-scale values, some of them even corresponding to the quantum scale processes, up to very large numbers corresponding to large scale processes in Nature.

Therefore, the algorithm presented has a very wide range of applications from the point of view of the output values it produces.

Furthermore, the CSN Algorithm could either be used to decrypt evolutionary fractals in Nature or to simulate random intelligence processes in the human brain.

It could also be used to replace the present A.I. classical models, or it could be used to add the missing procedural randomness to the present A.I. models, lacking this important property of the intelligence.

The CSN Algorithm can also be used as a function and operation-based encryption method.

Another possible use of the CSN Algorithm could be as a creative ideas generator. By replacing with knowledge, the functions in the cells of its corresponding CSN Matrix, one may be able to generate brand new ideas, such as those involved in scientific discoveries.

Thus, by placing knowledge or information in the cells of the CSN Matrix of Elements, and logical operations instead of mathematical operations in the cells of the CSN Matrix of Operations, the evolutionary fractal model generated by the algorithm will be able to generate new knowledge and science in a much more adequate manner than the model presented in [2].

The model presented in the paper above, will be thus seriously improved, since the respective model of generating knowledge proposed back in 2016, was a pretty simple one, linear and rather deterministic, and pretty much as primitive as the present A.I. models.

Another possible use of the CSN Algorithm is of being a starting point for other Enhanced Intelligence models (E.I. models), such as will be presented and described in other subsequent papers of the author on the same topic.

4 Discussion & Conclusions

After the decryption of the simplest evolutionary fractals, related to the most primitive life forms, e.g. such as plants, the second step will be to decrypt evolutionary fractals related to the more complex animal life forms.

At a further step, evolutionary fractals related to human beings, but not related to human intelligence, could be decrypted. An example for this could be the DNA mutations in humans, i.e. transition of the human DNA between its two different states after the exposure to a virus or after the exposure to the vaccine against the respective virus.

Another further development in using the CSN Algorithm could be to estimate and identify data in the cells of the CSN Matrices of elements and operations, at a certain stage in time, based on a known set of inputs/outputs.

Or, one may have data about both, inputs/outputs and some functions and operations in the cells of the CSN Matrices, and using this data, one may be able to find past or future stages of the CSN Algorithm.

Either way, to apply the CSN Algorithm to particular evolutionary processes, one should have prior specific knowledge of the respective field, in order to identify the most appropriate elements and operations of the CSN Matrices.

Finally, after decrypting the simpler evolutionary fractals above, the decryption of the human brain and of human brain processes may be then finally addressed.

The CSN Algorithm can also be used as a creative ideas generator, by placing information or knowledge instead of functions in the cells of its corresponding CSN Matrix, such as explained in the section Results of this paper.

This paper also proposes new approaches in the use and utilization of the huge computing power available nowadays, immense computing power which, among others, could be also employed to seek and find hidden secrets in Nature or in Nature's laws.

There is a rich literature regarding usual methods and algorithms involved in the A.I. But, since this newly introduced algorithm, the method presented and its related knowledge are at their beginnings, it is necessary that all who have followed this my research to be lenient and to show understanding regarding its lacks and shortcomings, but also to show a vivid gratitude to all its discoveries.

In a world that is basically and mostly market driven and profit driven, it may seem somehow farfetched to seek to generate a method and an algorithm to search and decrypt evolutionary fractals.

But, after all, as almost always, science and scientists are not working market driven or profit driven, but they are mainly working problem and solution driven in their undertakings.

And anyway, eventually, innovative business people almost always know how to profit, to monetize and to employ new generated science for their own purposes or their businesses, and in this case, this could be the encryption method associated with the CSN Algorithm.

Acknowledgement

My special thanks to Constanta Maritime University for the financial support offered throughout the years and now, by paying the fee for the publication of this paper, out of the research funds of the institution.

References

1. Nutu, C. S., Axinte T.: Communication with Nature based on a Combined Multiplicative/Additive Encryption Model, Advanced Topics in Optoelectronics, Microelectronics and Nanotechnologies XI, 124931H-10, Proceedings of SPIE, Vol. 12493, 2022: 369-378

2. Nutu, C. S., Axinte T.: Microelectronics and Nanotechnology and the Fractallike Structure of Information, Knowledge and Science, Advanced Topics in Optoelectronics, Microelectronics and Nanotechnologies VIII, 10010 100101I-1, Proceedings of SPIE, Vol. 10010, 2016: 394/402

3. Nutu, C. S.: Decryption of Evolutionary Fractals by means of Procedural Randomness generated with the CSN Matrix, International Journal of Computational Engineering Research (IJCER) ISSN 2250-3005, Vol.14, Issue 4, Jul-Aug 2024
4. Nutu, C. S., Axinte T.: Fractals. Origins and History, Journal of Marine Technology and Environment, Vol. 2, 2022, 48-51
5. M. P. Deisenroth, A. A. Faisal, C.S. Ong: Mathematics for Machine Learning, Cambridge University Press, 2020
6. Mitchel T.M.: Machine Learning, McGraw-Hill Science/Engineering/Math, 1997
7. Mehlig B.: Machine learning with neural networks, University of Gothenburg, Göteborg, Sweden 2021
8. Halvorsen H.P.: Python Programming, ISBN:978-82-691106-4-7, 2020
9. Thomas A.: An introduction to neural networks, 2019, https://coursehero.com
10. Ezekiel S., Pearlstein L., Alshehri A.A, Lutz A., Zaunegger J., Farag W.: Investigating GAN and VAE to Train DCNN, International Journal of Machine Learning and Computing, Vol. 9, No. 6, 2019, 10.18178, 774/781
11. Dumitriu A.: Istoria Logicii (The History of Logic), Editura Tehnica, Bucharest, 1993
12. Nutu C. S.: Hunting for Genius. What is Brilliance? Can A.I. be brilliant?, Journal of Marine Technology and Environment, Vol. 1, 2024, 29-34

Appendix:

The Python script corresponding to the CSN Algorithm

The Python programming language [8], is one of the most appropriate tools to show the actual application of the CSN Algorithm.

In the paper [1], the MATLAB software has been previously used for the same programing purposes, but the script in Python has more clarity and hence, the Python programing environment is also a more suitable one than MATLAB.

This Python script, in accordance with the particular form of the algorithm from above, is presented below, as an appendix to the body of the paper:

```
import numpy as np
import sympy
from sympy import*

#For the algorithm purposes, two CSN Matrices are used: CSN Matrix of elements
#and CSN matrix of operations

m=Symbol('m')
n=Symbol('n')
i=Symbol('i')
j=Symbol('j')
```

```
x=Symbol('x')

print('Enter the number of lines of the CSN Matrix')
m = int(input('Enter m>5:'))
print('Enter the number of columns of the CSN Matrix')
n = int(input('Enter n:'))

p=m*n;
print('The total number of possible Tkey values is:',p)

#Tkey is the random input value of the CSN Algorithm
Tkey = np.random.randint(0,p-1)
print('The Tkey value is: Tkey =',Tkey)

print('The number of first lines of functions f is:')
a = np.random.randint(1,m//3)
print('a=',a)

print('The number of next lines of functions g is:')
b = np.random.randint(1,m//3)
print('b=',b)

print('The number of next lines of functions h is:')
c = np.random.randint(1,m//3)
print('c=',c)

print('The number of last lines of functions i is:')
d = m-a-b-c
print('d=',d)

F = np.random.uniform(0,3, size=(a,m+n+1,n))
G = np.random.uniform(0,3, size=(b,m+n+1,n))
H = np.random.uniform(0,3, size=(c*n))
I = np.random.uniform(0,3, size=(d*n))

print('The matrix of coefficients of f polynomial functions is:')
print(F)
print()
print('The matrix of coefficients of g polynomial functions is:')
print(G)
print()
print('The matrix of coefficients of h logarithmic functions is:')
print(H)
print()
print('The matrix of coefficients of i exponential functions is:')
```

```
print(I)
print()

for j in range (a):
   def f(j):
      return F[j,:,:]
for j in range (b):
   def g(j):
      return G[j,:,:]
for j in range (a):
   print('The coefficients of polynoms f in line',j+1,'are:')
   print(f(j))
   print()
   def poly_f(j,x):
      return sum((a*x**(m+n+1-i) for i,a in enumerate(f(j))))
   print('The polynoms f in line',j+1,'are:')
   print()
   print(poly_f(j,x))
   print()

CSN_a_elements = np.array([poly_f(j,x)])
CSN_a_operations = np.array([Derivative(poly_f(j,x),x)])

print('The first "a" lines of the CSN Matrix of elements are:')
print(CSN_a_elements)
print()
print('The first "a" lines of the CSN Matrix of operations to transform the CSN Matrix
of elements are:')
print(CSN_a_operations)
print()

for j in range (b):
   print('The coefficients of polynoms g in line',a+j+1,'are:')
   print(g(j))
   print()
   def poly_g(j,x):
      return sum(a*x**(i+1) for i,a in enumerate(g(j)))
   print('The polynoms g in line',a+j+1,'are:')
   print()
   print(poly_g(j,x))

CSN_b_elements = np.array([poly_g(j,x)])
CSN_b_operations = np.array([Integral(poly_f(j,x),x)])

print('The second "b" lines of the CSN Matrix of elements are:')
```

```
print(CSN_b_elements)
print()
print('The second "b" lines of the CSN Matrix of operations to transform the CSN Matrix of elements are:')
print(CSN_b_operations)
print()

CSN_c_elements = np.array(H)
CSN_c_operations = np.log(CSN_c_elements)

print('The arguments of exponential functions h in line are:')
print()
print(CSN_c_elements)
print()
print('The third "c" lines of the CSN Matrix of operations to transform the CSN Matrix of elements are:')
print(CSN_c_operations)
print()

print('The arguments of logarithmic functions i in line are:')
print()
CSN_d_elements = np.array(I)
print(CSN_d_elements)
CSN_d_operations = np.exp(CSN_d_elements)

print('The last "d" lines of the CSN Matrix of elements are:')
print(CSN_d_elements)
print()
print('The last "d" lines of the CSN Matrix of operations to transform the CSN Matrix of elements are:')
print(CSN_d_operations)
print()

if (Tkey<a*n):
   CSN_Alg = np.array(CSN_a_operations[Tkey])
else:
      if (a*n<=Tkey<(a+b)*n):
            CSN_Alg = np.array(CSN_b_operations[Tkey-a*n])
      if ((a+b)*n<=Tkey<(a+b+c)*n):
            CSN_Alg = np.array(CSN_c_operations[Tkey-(a+b)*n])
      if ((a+b+c)*n<=Tkey<m*n):
            CSN_Alg = np.array(CSN_d_operations[Tkey-(a+b+c)*n])

print('The output of the CSN Algorithm is ',CSN_Alg)
```