Koch Infinite Fractal Curve Implementation for the Space Filling Problem in Manufacturing

Tanmay Debnath[a]

[a]PDPM Indian Institute of Information Technology, Design and Manufacturing Jabalpur, India

ABSTRACT:

Space filling problem has been one of the most intriguing problems in the domain of manufacturing science and engineering. Several approaches have been applied to solve the problem, ranging from infinite curves (Gospel Curve) to straight line approaches. This paper introduces a new approach of space filling using fractal curves. Fractal curves are defined as mathematical curves whose shape retains the same general pattern of irregularity, regardless of how much magnified the shape has been considered, basically the shape of a fractal. These curves are widespread in nature and hence using the same, a novel method has been prepared for generating a fractal curve to be implemented in the additive manufacturing process.

1. Introduction:

Fractal curves are considered as natural curves and have been utilised till now as a method to understand the complex mechanism of nature. Historically, Benoit Mandelbrot first introduced the concept of fractal, based on the definition of Hausdroff in 1919 [5]. He first recognised that many phenomena in nature are not actually solved by Euclidean Geometry and hence require special sets of rules and dimensions to define them. Fractal curves also found its way into the computer graphics and image processing segment, where using the same, micro-processing are done and high-power computations are performed. The fractal theory has been employed in different manufacturing processes to either understand the structural growth/formation or to produce fractal-like components for enhanced product performance [1]. Hence fractal theory does give a greater edge and a different angle of perspective for the observation of space-filling as a potential problem to solve. There have been several instances of research ideas present all over the internet claiming to solve the problem of space-filling in a novel way. Here only those are chosen which has a specific central idea concerning the space-filling from fractal perspectives.

2. Literature Review:

In [2], Paul Bourke has developed a space filling algorithm for procedurally generating a range of digital assets including 2 dimensional textures and 2.5-dimensional texture roughness. Here, the authors have tried to present an algorithm for space filling and texture generation using fractals. The procedural and iterative properties of this algorithm make it ideal for creating textures and geometry at sufficient detail to meet frame rate dictated performance constraints within a gaming engine dependent on the players distance and viewing direction.

The implementation of fractal curves has several advantages – the hypothesis has been confirmed by [3]. They have discussed three methods for the implementation of FASS (Space-filling, self-avoiding, simple and self-similar) curve for arbitrary boundaries. They have trimmed the generated fractals along the boundary by CAM package. But the trimmed FASS path seems to be losing its features at those boundary points. They used HFM (Hybrid Fractal Toolpath) which generates the hybrid toolpath, combination of fractal and raster paths.

In this paper, the problem has been addressed using similar process as described in [3]. But [3] proposed the usage of Gospel curve for the filling of the curves and in this paper, the Koch curve method has been used. Koch curve gave the flexibility to extend the algorithm vertically and provided the flexibility to decide the space to be kept in between, based on the filament material and thickness of the nozzle. This flexibility would be advantageous for industries which keep on experimenting with various models of variant complexities. Being of specific dimension and scale the curve can be extended to any dimension, as per the wish of the user. This paper will show the implementation on 2D plane as well as when the model is considered in a 3D frame. As an extension, in the software produced, there is a section for the generation of G-Code. The program would automatically calculate the generated points in the space and would then convert into an editable code, which can be directly implemented into the 3D Printer or CNC Milling Centre Machine for the process to start.

3. <u>Koch Curve Method:</u>

The Concept of Koch Curve has been published back in 1904 by the famous Swedish mathematician Fabian Helge von Koch in his article 'snowflake curve', generated from an equilateral triangle. Each side is trisected and the centre segment replaced by two sides of a smaller equilateral triangle projecting outward. The process is repeated *ad infinitum* [4]. The method generated by him is based upon the concept of snowflakes and they have been defined mathematically as fractal curves. In this paper, the simplest version of the Koch curve has been used, in the form of a straight line.

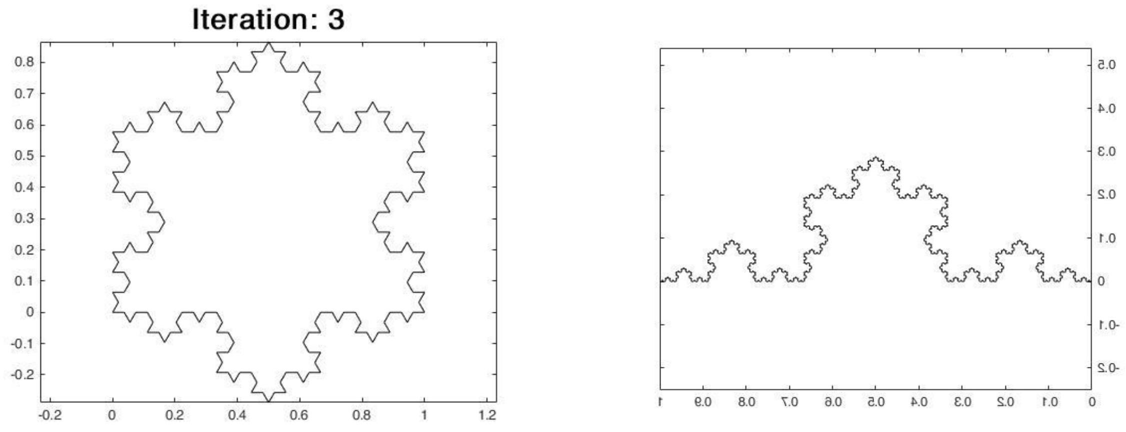


Figure 1: (a)Koch Snowflake Curve (for n=3). (b) Koch Line Curve (for n=4)

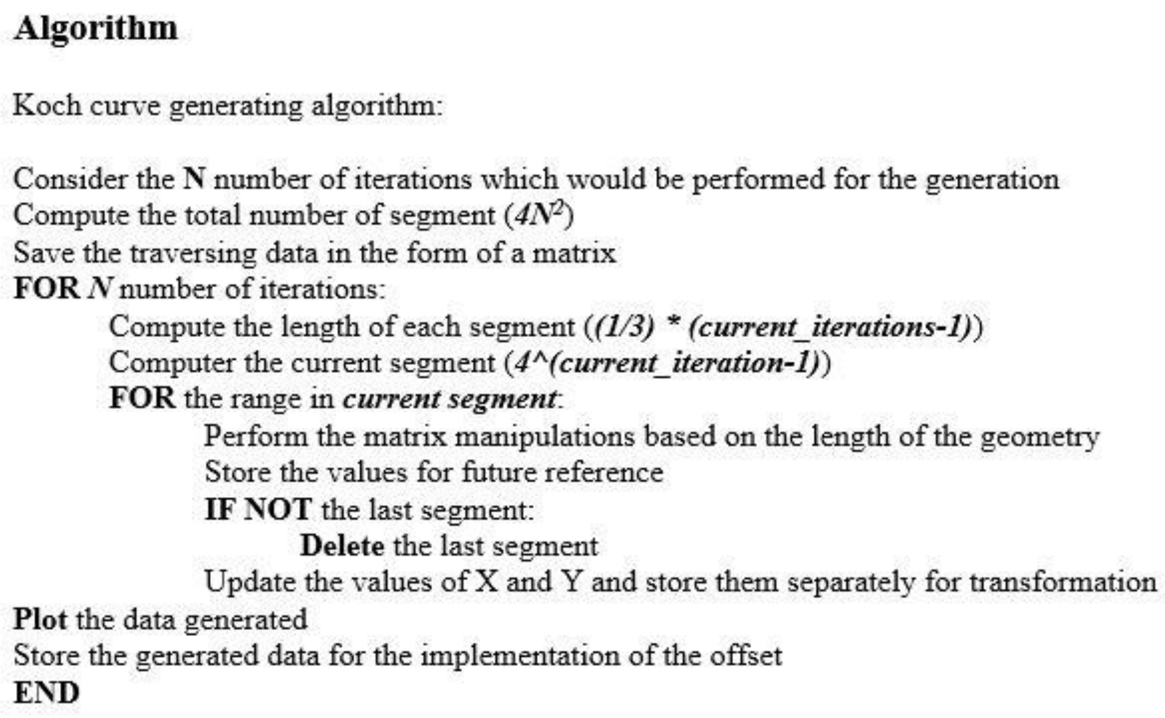
Figure 1, completely describes the used curve in modelling the implemented form of the curve. It has been observed that if the curve was programmed for higher number of generations then the nozzle of the 3D Printer considers the point to be too close to each other and hence doesn't differentiate in between the points. This creates a problem in the accuracy of the to-be modelled part. Upon experimentation it has been observed that a generation of 2 (n=2) is sufficient for modelling the entire part, of any desired dimension. Figure 1(a) shows the entire geometry of the snowflake that can be mathematically derived from the Koch equation. But it is not required in the following context because as the snowflake reduces in size the control points become inconsistent to the dimensions of the nozzle of the machine. Hence the line algorithm has been chosen here for the same purpose.

4. <u>Algorithm:</u>

In the Koch line algorithm, for 0th generation (n=0), a single line has been considered and using the same as the base all the further applications have been performed. Firstly, the line has been trisected, at points 1/3rd and 2/3rd of the total length of the arbitrary line. The dimension of the line is controllable via the program. In between the points 1/3 and 2/3, an equilateral triangle is placed of edge length 1/3 of the total length of the line. The height of the triangle in x-direction would move up to 0.5 of the total line segment. After the first iteration, the total number of individual segments would count up to 4. After each generation the total number of individual line segments generated would be guided by the following equation:

$$x = 4^n \quad \left| \begin{array}{l} \forall n \in N \\ n \geq 0 \end{array} \right.$$

Where, x is the total number of segments and n is the number of generations we are considering. Computationally, the greater number of generations we are considering the program becomes more expensive. It might be unfitting for systems with low graphics and computation power. Hence considering generalised power of computation for all the systems, a generation of 2 has been considered.



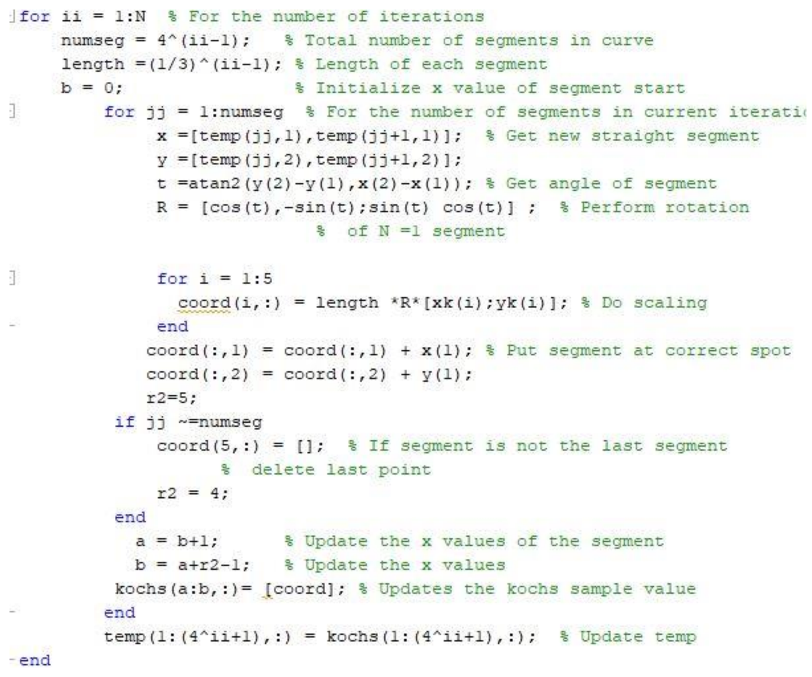
Algorithm

Koch curve generating algorithm:

```
Consider the N number of iterations which would be performed for the generation
Compute the total number of segment ($4N^2$)
Save the traversing data in the form of a matrix
FOR N number of iterations:
        Compute the length of each segment ((1/3) * (current_iterations-1))
        Computer the current segment (4^(current_iteration-1))
        FOR the range in current segment:
                Perform the matrix manipulations based on the length of the geometry
                Store the values for future reference
                IF NOT the last segment:
                        Delete the last segment
                Update the values of X and Y and store them separately for transformation
Plot the data generated
Store the generated data for the implementation of the offset
END
```

Figure 2: Algorithm defining the process of generating the Koch curve

From the above algorithm definition, it is clear that the space complexity of the program is O(n^2). The MATLAB implementation of the same has been depicted afterwards. However, in the MATLAB algorithm, the matrix manipulation system has been used because of the extensive dimensionality expansion capacity and manipulability. Hence for the same, the matrix rotation and translation methods have been implemented.



```
for ii = 1:N  % For the number of iterations
    numseg = 4^(ii-1);    % Total number of segments in curve
    length =(1/3)^(ii-1); % Length of each segment
    b = 0;                % Initialize x value of segment start
        for jj = 1:numseg  % For the number of segments in current iterati
             x =[temp(jj,1),temp(jj+1,1)];  % Get new straight segment
             y =[temp(jj,2),temp(jj+1,2)];
             t =atan2(y(2)-y(1),x(2)-x(1)); % Get angle of segment
             R = [cos(t),-sin(t);sin(t) cos(t)] ;  % Perform rotation
                            %  of N =1 segment

             for i = 1:5
               coord(i,:) = length *R*[xk(i);yk(i)]; % Do scaling
             end
            coord(:,1) = coord(:,1) + x(1); % Put segment at correct spot
            coord(:,2) = coord(:,2) + y(1);
            r2=5;
         if jj ~=numseg
             coord(5,:) = [];  % If segment is not the last segment
                   %  delete last point
             r2 = 4;
         end
           a = b+1;      % Update the x values of the segment
           b = a+r2-1;   % Update the x values
         kochs(a:b,:)= [coord]; % Updates the kochs sample value
        end
        temp(1:(4^ii+1),:) = kochs(1:(4^ii+1),:);  % Update temp
end
```

Figure 3: MATLAB implementation of the above stated algorithm

The above code generates the Koch line curve for a given number of generations. Here 2 generations have been considered. So, the total number of line segments that would make up a single Koch line would be 16 in number.

5. Implementation:

The single line of the Koch curve has been implemented into a space and has been extended till the breadth and height of the model segment in 2D shape. However, there would be chances of self-intersection and hence to stop the same from happening the line has been programmatically corrected where for each iteration in the height-wise direction the program would automatically detect the self-intersecting parts and would remove the extended parts and combine them as a single line. This reduces the obscurity of the nozzle to move in unprecedented directions, causing a hassle.

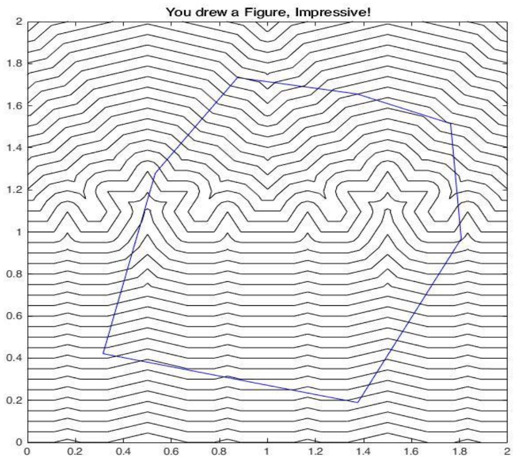


Figure 4: An arbitrary shape augmented into the range of the Koch line curve

It is observable that after some point of time, in the extension phase, there has been a decline in the features of the Koch line curve. This causes to an advantage because this ensures the continuity of the product in its dimensions and also can be used to design intricate structures. The task would now be to cut the Koch line segments till the part of the arbitrary shape and align them accordingly such that the program data can be stored and hence can be implemented for the code generation process.

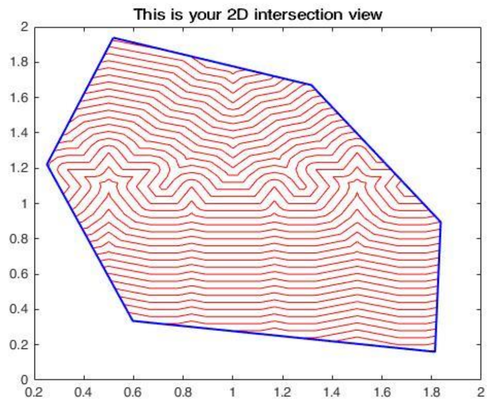


Figure 5: The processed and segmented out arbitrary figure

As depicted, the model shown here is the intersection of the generated Koch curve and the drawn arbitrary shape. For clarification the arbitrary shape has been shown in 'blue' colour while the Koch curve has been depicted in 'red' lines. The cut lines became somewhat discreet and hence needed to be processed. For the same, they were connected to the immediate next line end point thus making the process continuous for the nozzle or the machining centre drill-bit.

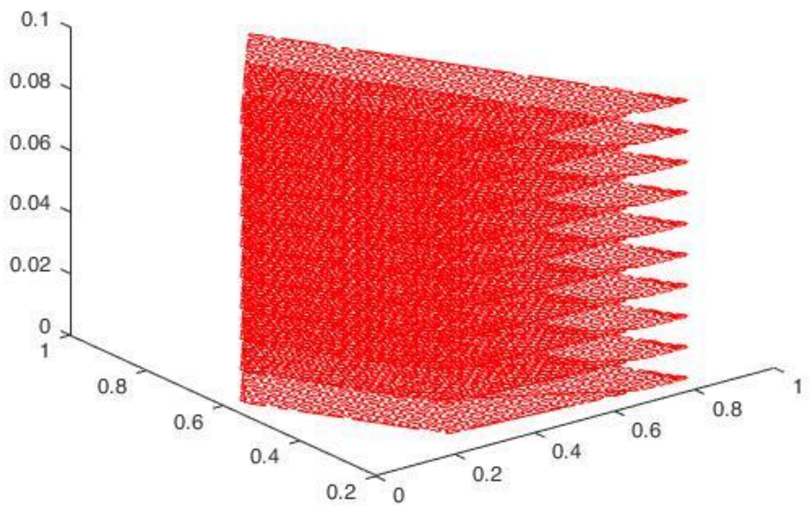
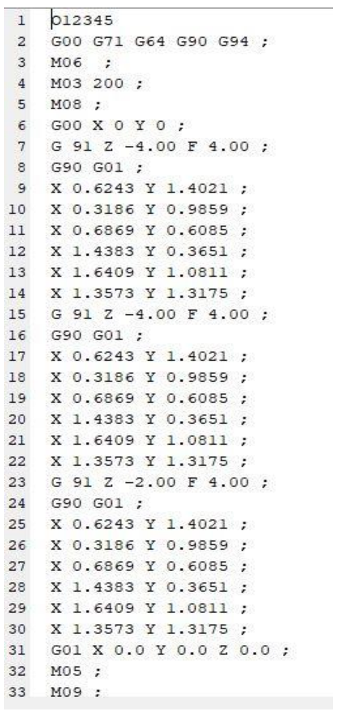


Figure 6: The 3D Printing stacked over a particular plane

6. G-Code Generation:

The main function of the program is to generate the printable or editable G-Code which would be available for the direct usage or as per the needs of the users. The G-Code generating segment takes the input of all the generated points in the plane, stored in the RAM of the system. Based on sequencing, the software has been designed to produce the G-Code for each operation precisely.



```
012345
G00 G71 G64 G90 G94 ;
M06  ;
M03 200 ;
M08 ;
G00 X 0 Y 0 ;
G 91 Z -4.00 F 4.00 ;
G90 G01 ;
X 0.6243 Y 1.4021 ;
X 0.3186 Y 0.9859 ;
X 0.6869 Y 0.6085 ;
X 1.4383 Y 0.3651 ;
X 1.6409 Y 1.0811 ;
X 1.3573 Y 1.3175 ;
G 91 Z -4.00 F 4.00 ;
G90 G01 ;
X 0.6243 Y 1.4021 ;
X 0.3186 Y 0.9859 ;
X 0.6869 Y 0.6085 ;
X 1.4383 Y 0.3651 ;
X 1.6409 Y 1.0811 ;
X 1.3573 Y 1.3175 ;
G 91 Z -2.00 F 4.00 ;
G90 G01 ;
X 0.6243 Y 1.4021 ;
X 0.3186 Y 0.9859 ;
X 0.6869 Y 0.6085 ;
X 1.4383 Y 0.3651 ;
X 1.6409 Y 1.0811 ;
X 1.3573 Y 1.3175 ;
G01 X 0.0 Y 0.0 Z 0.0 ;
M05 ;
M09 ;
```

Figure 7: The G-Code generation program generated a demo program

Based on the similar design of the G-Code, a demo model has been prepared. The model contains the exact G-Code generated by the program. Based on the same, the GUI has been designed which would accumulate all the functionalities and would eventually control all the aspects of the program.

7. **Output:**

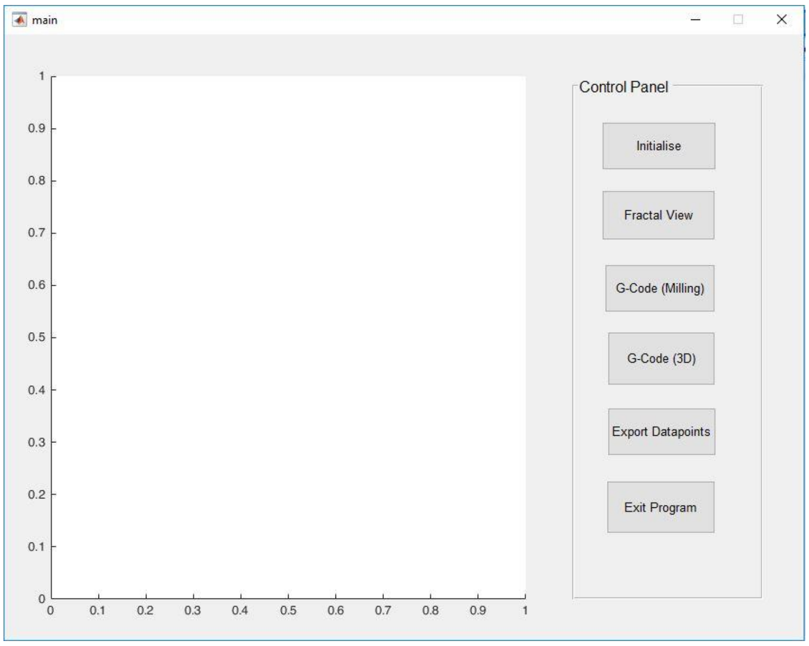


Figure 8: The GUI of the depicted program

The GUI covers all the aspects of the program and hence adheres to the user effectiveness. As depicted above, the GUI contains interactive buttons, working as a control panel. The 'Initialize' – button helps in the initialisation of the program. After initialisation, the user would be able to draw in the given space. Since this is the experimental version, hence there has not been any provisions made for the slicing of a 3D geometry for making the entire geometry. After the drawing has been made, the user is supposed to press the 'Fractal View' button – which creates the fractal view of the created diagram in the space. After that, dual choice has been given, where the user can use the same program to generate the Milling operations or Additive Manufacturing operations. Provisions have been made for the user to implement their own code and hence the code is stored in an XLS file.

8. **Result:**

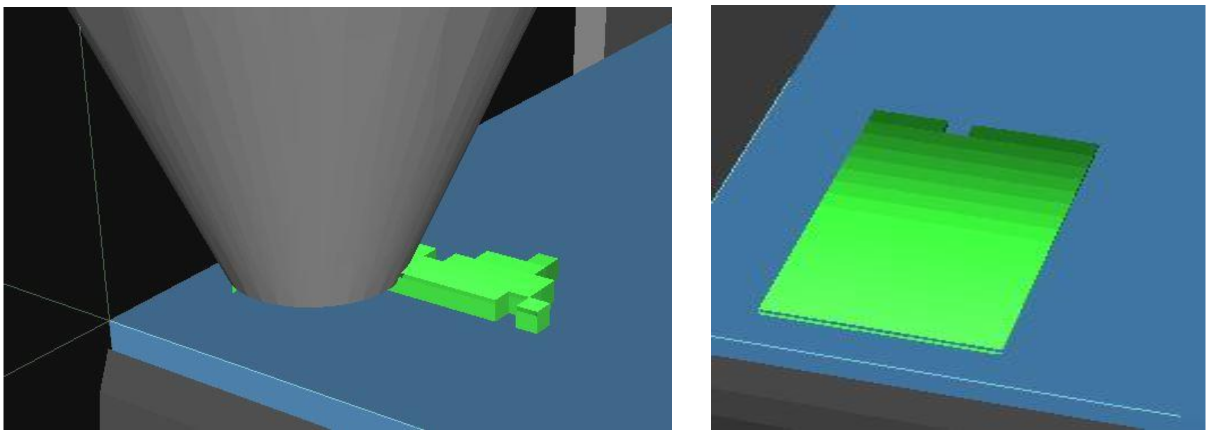


Figure 9: Prototyping in the designed framework

The following framework has been generated using the CNC Simulator Pro software and hence using the same, a virtual prototype of the working frame has been prepared.

REFERENCES:

1. Mwema, F. M., Akinlabi, E. T., Oladijo, O. P., Fatoba, O. S., Akinlabi, S. A., & Tălu, S. (2020). *Advances in manufacturing analysis: fractal theory in modern manufacturing. Modern Manufacturing Processes, 13–39.* doi:10.1016/b978-0-12-819496-6.00002-6
2. Bourke, Paul. (2013). A Space Filling Algorithm for Generating Procedural Geometry and Texture. GSTF Journal on Computing. 3. 10.7603/s40601-013-0004-2.
3. Kapil, Sajan & Joshi, Prathamesh & Vithasth, Hari & Rana, Dhirendra & Kulkarni, Pravin & Bhagchandani, Ranjeet & K.P., Karunakaran. (2016). Optimal space filling for additive manufacturing. Rapid Prototyping Journal. 22. 660-675. 10.1108/RPJ-03-2015-0034.
4. Kemp, M. Science in culture. *Nature* **444,** 1008 (2006). https://doi.org/10.1038/4441008a
5. Lokenath Debnath * (2006) A brief historical introduction to fractals and fractal geometry, International Journal of Mathematical Education in Science and Technology, 37:1, 29-50, DOI: 10.1080/00207390500186206