# Enhancing Computational Thinking with Spreadsheet and Fractal Geometry: Part 2 Root-finding using Newton Method and Creation of Newton Fractals

K.P Soman, Manu Unni V.G, Praveen Krishnan, V. Sowmya
Centre for excellence in Computational Engineering and Networking (CEN),
Amrita Vishwa Vidyapeetham, Coimbatore, Tamil Nadu, India

## ABSTRACT

This article shows how Newton's iterative methods for finding root of a polynomial equation can be used to create fractals in spreadsheets. Newton's method has served as one of the most fruitful paradigms in the development of complex iteration theory. The process of iteration is impossible to carry out by hand but extremely easy to carry out with a computer. By doing such experiments students get a feeling that they have the power to explore the uncharted wilderness of the dynamics of Newton's method. It gives mathematics an experimental component. It also illustrates a symbiotic relationship between technology and mathematics [1]. Technology can be used to develop our intuition, and mathematics is used to prove that our intuition is correct. The article explores Innovative use of Microsoft Excel's What-if Analysis tool to do automation of repeated computation. The method employed can also be used for Neural Network training and data clustering [9] in Excel. A wide variety of fractals can be created by using different polynomial equations [2-7].

**Keywords**— Fractal, Newton method, Spreadsheet, Fractal geometry.

## 1. INTRODUCTION

In mathematics one come across systems of equations quite frequently. Unfortunately, many of them cannot be solved by algebraic manipulation. one therefore need to find a way to solve the equations numerically, in the hope of gaining as much accuracy as possible. There is a simple technique in elementary calculus known as Newton's method. Newton's method is a method for iteratively approximating the root of an equation $f(x)=0$ using first derivative alone. In other words, we want to find a value $x^*$ such that $f(x^*)=0$. One may proceed by giving an "educated" guess and then refining the guess over and over again. If the function is quite simple, one may be able to do some algebra to find a root $x$ exactly. For instance, if $f(x)$ is a quadratic polynomial, he or she can use the quadratic formula. If $f(x)$ is a polynomial of degree 3 or 4, there are messier formulas which work as well. But if $f(x)$ is a higher degree polynomial or an even more complicated function, there is no analog to the quadratic formula, i.e. there is no systematic process to algebraically determine the roots exactly. One has to approximate, and Newton's method is just one way of doing this.

Consider the graph showing $x$ versus $f(x)$. We need to find $x$ for which $f(x)=0$. Or in other words, one need to find value of $x$ at which the curve crosses x-axis. Let that value of x be $x^*$. Newton algorithm proceeds as follows. One can make a guess about $x^*$. Let the guessed value be $x_0$. Since $f(x)$ is given, one can immediately find $f(x_0)$ and derivative of $x$ at $x_0$ i.e., $f'(x_0)$). We know, $f'(x_0)$ is the slope of the tangent at the point $(x_0, f(x_0))$. Let this tangent meet the x-axis at $x=x_1$. The coordinates of the meeting point is thus $(x_1,0)$. Notice that, the slope of the line joining the points $(x_0, f(x_0))$ and $(x_1,0)$ is $f'(x_0)$. That is

$$f'(x_0)=\frac{0-f(x_0)}{x_1-x_0}$$

$$x_1-x_0=\frac{-f(x_0)}{f'(x_0)} \text{ or } x_1=x_0-\frac{f(x_0)}{f'(x_0)}$$

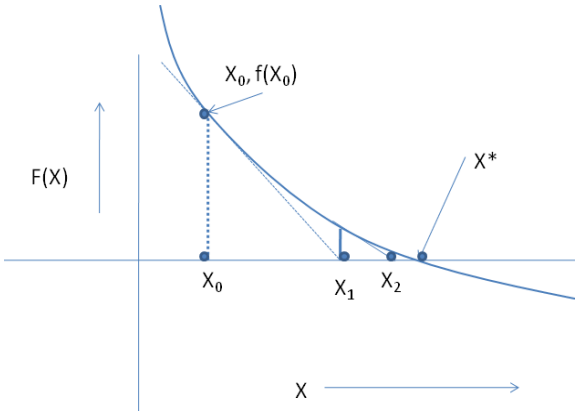



**Fig 1: Newton Method of finding roots- A graphical view**

Take $x_1$ as the next approximate solution. Next find $x_2$ using the formula

$$x_2=x_1-\frac{f(x_1)}{f'(x_1)}$$

One can repeat this process till it converges to $x^*$ with desired precision

The iterative formula is thus

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad (1)$$

If f(x) crosses x-axis several times, depending on the initial guess about the root, the algorithm converges on one of the roots (the point where curve crosses x-axis).

## 2. COMPLEX ROOTS OF A POLYNOMIAL EQUATION

The question arises as what is going to happen if one now allow x (and f(x)) to be complex numbers rather than just real numbers. So the domain of function is a complex plane. Can one get complex roots?. This question makes perfect sense since a polynomial of degree n has n roots, which is a "Fundamental Theorem of Algebra". It is also possible to define the derivative over complex numbers just as for real numbers, so formula (1) makes sense. It turns out that polynomials are all complex-differentiable, and the complex-derivative is (surprise!) the same as the real-derivative. So formula (1) still holds, even though the geometry is much more complicated. (In part since x is now in some sense 2-dimensional and likewise f(x), so one need to think in 4 dimensions!)

### *A. Newton Fractal*

Newton fractal is obtained by applying Newton's iterative method for finding roots to a fixed complex polynomial $f(z)$ where $z = x + iy$

Newton's iterative method for a polynomial equation $f(z) = 0$ is given by

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)}$$

Each point in the complex plane will be associated with a root of the polynomial because based on starting point, algorithm converge on one of the roots. For example, $f(z) = z^3 - 1 = 0$ have three roots in the complex plane as shown in figure 2. The set of all starting points $z_0$ in the complex plane which are getting converged to a particular root form a region with complexpattern. A starting point $z_0$ in the complex plane when selected and applied as a starting point in Newton's iteration, yields a sequence of points $z_1, z_{2,\ldots}$ which finally converge on one of the roots. It is usually difficult to tell to which root the algorithm will converge, given a starting point. We use a color to distinguish the set of all starting points which on iteration converge on a particular root. Number of sets in the plane is equal to the number of roots the polynomial has. It is found that the color-tagged-points are fractal in nature. Here we demonstrate an example of drawing Newton fractal for a polynomial of order three in excel. For the polynomial as $f(z) = z^3 - 1$, the figure looks like the Fig.3 given below. Wide variety of coloring-scheme is possible.

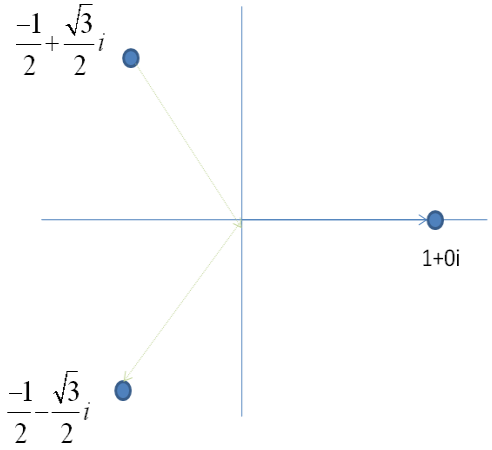



**Fig:2 Three roots of** $f(z) = z^3 - 1 = 0$

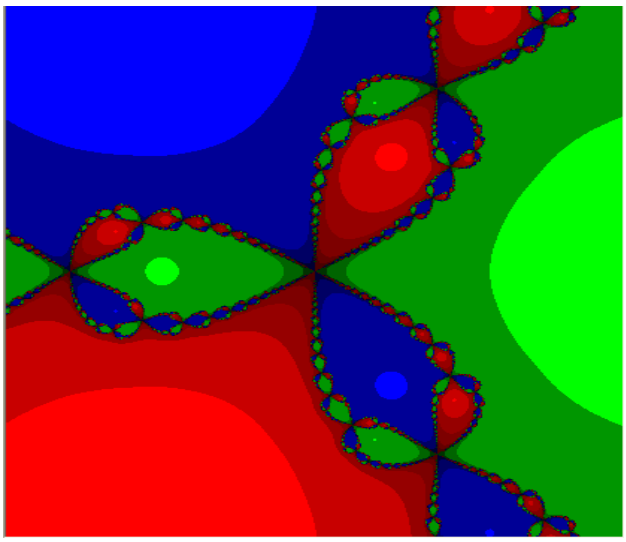


**Fig: 3 Newton Fractal for** $z^3 - 1 = 0$

One can 'grade' the color according to number of iteration taken to converge. So a dark red point indicates fast convergence to root corresponding to Red and light red indicate large number of iteration to converge. This reveals full dynamics of Newton method of iteration to a student. From figure 3 it can be easily inferred that there are certain regions in the complex domain where a slight change in the starting will lead to a different solution (a different root). The pattern is very intricate. Therefore this picture serves as a glimpse of behavior of complex dynamical systems whose evolution is described by iterative equations like Newton's iterative step.

## 3. IMPLEMENTATION STRATEGY IN EXCEL WITH THE USE OF WHAT-IF-ANALYSIS

One of the skills that a student must learn is innovative use of tools or concepts in problem solving. The methodology used in this article is one such innovation. There are three kinds of what-if analysis tools that comes with Microsoft Excel: scenarios, data tables, and Goal Seek. Scenarios and data tables take sets of input values and determine possible results. A data table works only

with one or two variables, but it can accept many different values for those variables. A scenario can have multiple variables, but it can accommodate only up to 32 values. Goal Seek works differently from scenarios and data tables in that it takes a result and determines possible input values that produce that result. In this application, we used only 'data table' option of the what if analysis.

Essentially it works as follows. For one value of x and y representing the complex number $z_0$ = x+iy, one show how computation is to be done and decide on which root the computation converges on. Based on this example, what-if analysis do the computation and decision for other combinations of x and y or in other words other values of $z_0$ – the starting point of iteration. Considering the position of roots of $f(z) = z^3 - 1$, take a domain with x and y-axis ranging from -2 to +2 . Note that y-axis is imaginary axis. So the top- left and bottom-right corners are thus -2+2i and 2-2i respectively. Coordinate-wise those points are (-2,2) and (2,-2) There are infinite number of points in our domain. To draw the fractal quickly, one can take a moderate sized rectangular grid in the domain say 64 by 64 and compute the value of color for each grid point (which act as $z_0$ ) that give a coarse resolution picture of the fractal. The implementation strategy uses following facilities in Excel.

1) A rectangular array of excel cells themselves can act as our points in the domain. That's is a 64 by 64 block of cells act as our grid points
2) The size of the excel cells can be reduced to extremely small size so that it can act as a pixel or a point. Note that excel allows coloring of the cells. This can be easily done using conditional formatting.
3) What-If analysis facility will take care of repetitive nature of computation. We need to show only how one iteration works.

Now look at the nature of computation involved in drawing Newton fractal.

**Computation involved**

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)}$$

Where $f(z_n) = (z_n^3 - 1);\ f'(z_n) = 3z_n^2$

Putting $z_n = x_n + iy_n$, and on simplification

$$x_{n+1} = \frac{2}{3}x_n + \frac{1}{3}\frac{\left(x_n^2 - y_n^2\right)}{\left(\left(x_n^2 - y_n^2\right)^2 + 4x_n^2 y_n^2\right)} \quad ;$$

$$y_{n+1} = \frac{2}{3}y_n - \frac{2}{3}\frac{x_n y_n}{\left(\left(x_n^2 - y_n^2\right)^2 + 4x_n^2 y_n^2\right)}$$

Taking $d = 3*\left(\left(x_n^2 - y_n^2\right)^2 + 4x_n^2 y_n^2\right)$, the expressions reduces to

$$x_{n+1} = \frac{2}{3}x_n + \frac{x_n^2 - y_n^2}{d} \quad (2)$$

$$y_{n+1} = \frac{2}{3}y_n - 2\frac{x_n y_n}{d} \quad (3)$$

Iterate this 20 times assuming in 20 steps, iterated value reaches near one of the roots. The next step is to find out to which root the current iterated value is nearby. This is accomplished as follows. Let $(x_{20}, y_{20})$ be real and imaginary part of the iterated value after 20 iterations.

Let root No 1 be (1, 0) . Then square of distance to this root from $(x_{20}, y_{20})$ is $(x_{20} - 1)^2 + y_{20}^2$

Let root No 2 be $\left(\frac{-1}{2}, \frac{\sqrt{3}}{2}\right)$. Then square of distance to this root from $(x_{20}, y_{20})$ is $\left(x_{20} + \frac{1}{2}\right)^2 + \left(y_{20} - \frac{\sqrt{3}}{2}\right)^2$

Let root No 3 be $\left(\frac{-1}{2}, \frac{-\sqrt{3}}{2}\right)$. Then square of distance to this root from $(x_{20}, y_{20})$ is $\left(x_{20} + \frac{1}{2}\right)^2 + \left(y_{20} + \frac{\sqrt{3}}{2}\right)^2$

Once we obtain the square of distances, the index of the minimum value will point to which root, $(x_{20}, y_{20})$ is close by. If it is closest to root No 1, we tag the starting point of the iteration as 1 etc.

Repeat this procedure with coordinates of each point in the grid as starting values of iteration. Thus each grid point will be assigned an integer number from the set {1, 2, 3}.

Experiment No: 1: Draw Newton fractal for $z^3 = 1$

Step 1: In cells A1 to A4 enter the strings "Iteration No", "Denominator", "x-coordinate", "y-coordinate".

Step 2: In B3 and B4 enter 1. This represent our starting x and y coordinate (or as a complex number $z_0 = x + iy$ =1+1i.

Step 3: In C1 enter 1 and in D1 enter 2. Select and drag till V1. This series of numbers represent iteration index.

Step 4 :In C2 enter the formula for denominator 'd' based on the x and y values in B3 and B4 . The formula is

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Iteration No | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 2 | denominator | | 12.0 | 1.4 | 0.4 | 8.2 | 3.2 | 2.9 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| 3 | x-cordinate | 1.0 | 0.7 | 0.6 | 1.2 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| 4 | y-coordinate | 1.0 | 0.5 | -0.1 | 0.3 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

**Fig:4 Snapshot of excel sheet for Experiment**

=3*((B3^2-B4^2)^2+4*(B3^2)*(B4^2)). This corresponds to $d = 3*\left(\left(x_n^2 - y_n^2\right)^2 + 4x_n^2 y_n^2\right)$.

Step:5 In C3 enter the formula for update of the x value of new iteration .The formula is =(2/3)*B3+(B3^2-B4^2)/C2. This corresponds to $x_{n+1} = \frac{2}{3}x_n + \frac{\left(x_n^2 - y_n^2\right)}{d}$

Step:6 In C4 enter the formula for update of the y value of new iteration .The formula is =(2/3)*B4 - 2*B3*B4)/C2. This corresponds to $y_{n+1} = \frac{2}{3}y_n - \frac{2x_n y_n}{d}$

Step:7. Select cells C2:C4 and drag fill handle till V4. The resulting values in these cells represent denominator, x and y values for 20 iterations. Our interest is only in the x and y values at end of 20th iteration. The excel sheet with some formatting look as shown in Fig.4

Thus initial point (1,1) used in newton iteration converged on (1,0) after 20 iteration. It converged on a root . Next we have to certain to which root (i.e., index of the root) it is converged or nearby.

Step 8:Precompute the value of $\sqrt{3}/2$ in cell Z2 for further use.

Enter the string 'root of 3 by 2' in cell Z1 and =sqrt(3)/2 in cell Z2.

Step 9: Enter the three root co-ordinates in cells AB2 to AB6 (See the figure below)

In cell AB1 enter string "x-roots" and in AC1 enter "y-roots"

In cell AB2 type 1 andinAC2type 0. These values represent x-y coordinate of first root

In cell AB3 type-0.5 and in AC3 type =Z2. These values represent x-y coordinate of second root

In cell AB4 type -0.5 and in AC4 type = -Z2. These values represent x-y coordinate of third root

Step 10: Computation of distance of the final iterated x-y value (point) obtained in cells V3 and V4 from the three roots

In cell X3 enter string "dist-root1" and drag to X5

In cell Y3enter = ($V$3-AB2)^2+($V$4-AC2)^2 and drag till Y5

Step 11: Computing minimum of distances to the three roots

Enter =min (Y3:Y5) in Y6

Step 12: Computing index of the root to which iterated value is near by

Enter =if(Y6=Y3, 1 , if (Y6=Y4, 2,3)) in Y7

This completes one iteration of newton method for one starting point. Y7 contains the index of the root to which finally iteration converged starting from a given point given in cells B3 and B4.

| | X | Y | Z | AA | AB | AC |
|---|---|---|---|---|---|---|
| 1 | | | root of 3 by 2 | | x-roots | y-roots |
| 2 | | | 0.866 | | 1.0 | 0.0 |
| 3 | dist -root1 | 0.0 | | | -0.5 | 0.866 |
| 4 | dist -root2 | 3.0 | | | -0.5 | -0.866 |
| 5 | dist -root3 | 3.0 | | | | |
| 6 | min value | 0.0 | | | | |
| 7 | index | 1 | | | | |

**Fig: 5 Another Snapshot of excel sheet for Experimentation**

Step13: Prepare Table for What-if Analysis

In Cell B8 and C8 enter -2 and -1.9 respectively. Select these two cells and drag the formula till BN8

In Cell A9 and A10 enter -2 and -1.9 respectively. Select these two cell and drag till A73

In Cell A8 (the corner cell of the table) enter =Y7 (this corresponds to index of the converged root in the example iteration we have shown See step 12.)

Step 14: Use of What if Analysis for computing index of the roots to which algorithm converges for different combination of x-y coordinates given by the Data-Table.

Select Cells A8:BN73 and go for what-If Analysis. Choose Row-input cell as B3 and Column-input cell as B4. The output looks as shown in Fig.6.

Step 15: Select the cells in tables starting from B9 and ending in BN73 and go for conditional formatting (available at the 'home' Tab in Excel-2010) and choose any one of the three color palette. A part of the excel sheet look like as shown in the Fig.7

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Iteration No | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2 | denominator | | 12.0 | 1.4 | 0.4 | 8.2 | 3.2 | 2.9 | 3.0 |
| 3 | x-cordinate | 1.0 | 0.7 | 0.6 | 1.2 | 1.0 | 1.0 | 1.0 | 1.0 |
| 4 | y-coordinate | 1.0 | 0.5 | -0.1 | 0.3 | 0.1 | 0.0 | 0.0 | 0.0 |
| 5 | | | | | | | | | |
| 6 | | | | | | | | | |
| 7 | | | | | | | | | |
| 8 | 1.0 | -2.0 | -1.9 | -1.9 | -1.8 | -1.8 | -1.7 | -1.6 | -1.6 |
| 9 | -2.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| 10 | -1.9 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| 11 | -1.9 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| 12 | -1.8 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| 13 | -1.8 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| 14 | -1.7 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| 15 | -1.6 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |

**Fig:6 Snapshot of excel sheet illustrating What-if Table**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 34 | -0.4 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| 35 | -0.4 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| 36 | -0.3 | 2.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| 37 | -0.3 | 2.0 | 1.0 | 3.0 | 2.0 | 2.0 | 3.0 | 3.0 | 3.0 |
| 38 | -0.2 | 2.0 | 3.0 | 1.0 | 3.0 | 1.0 | 2.0 | 3.0 | 3.0 |
| 39 | -0.1 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 3.0 | 3.0 |
| 40 | -0.1 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 3.0 |
| 41 | 0.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| 42 | 0.1 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 2.0 |
| 43 | 0.1 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 2.0 | 2.0 | 2.0 |
| 44 | 0.2 | 3.0 | 2.0 | 1.0 | 2.0 | 1.0 | 3.0 | 2.0 | 2.0 | 2.0 |
| 45 | 0.3 | 3.0 | 1.0 | 2.0 | 3.0 | 3.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| 46 | 0.3 | 3.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| 47 | 0.4 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| 48 | 0.4 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| 49 | 0.5 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| 50 | 0.6 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |

**Fig:7 Snapshot of excel sheet illustrating conditional formatting**

Step:16 Reduce the cell Size

Select columns A to BN, right click the mouse and choose Colum width. Give column width as 0.1.

Select Rows 1 to 73, right click the mouse and choose Row height. Give Row-height as 1. The output looks like as shown in Fig.8.

Step: 17.Select Cells A8:BN73 and go for surface plot. From the plot remove unwanted objects like axis, grids and labels. Select the plot area, Right click the mouse and go for 3D rotation. For rotation about x axis as 15 degree, rotation about y-axis as 90 degree and perspective as 15 degree, the plot look as in Fig.9

Step: 18 Enjoy the picture for a while and go for making a fractal of size 1024 by 1024.

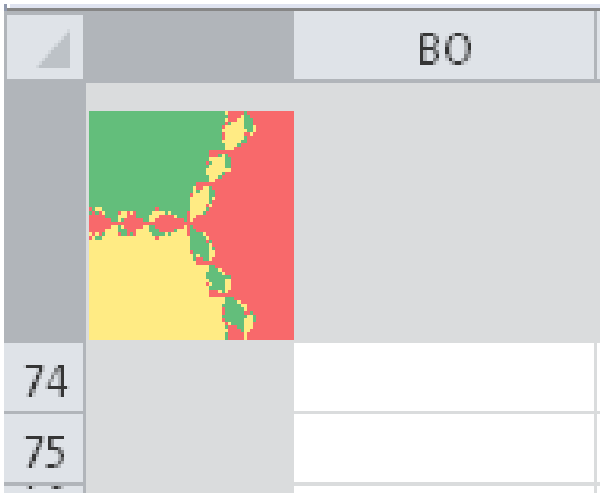



**Fig.8 Output of Experiment with size-reduced cells**

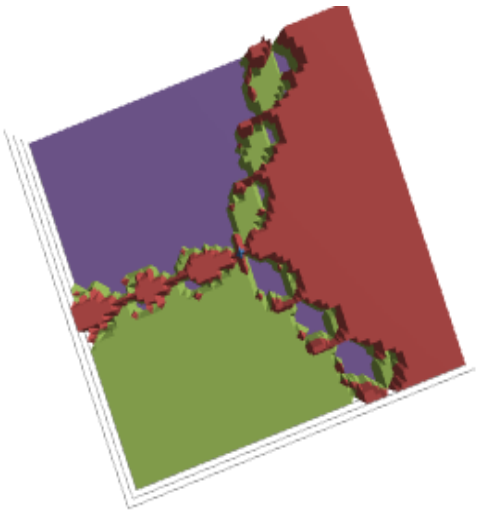


**Fig: 9 Surface plots of the data**

*A. Drawing Newton Fractal with VBA*

Subroutine Macro1 () is the VBA code required for plotting Newton fractal for $Z^3-1=0$.

In the macro, the following code shrinks the cell size to that of pixel

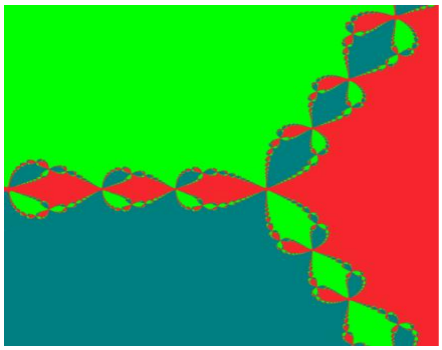
```
Cells.Select
Selection.ColumnWidth = 0.1
Selection.RowHeight = 1
```

The remaining part of the code is eslf explanatory.

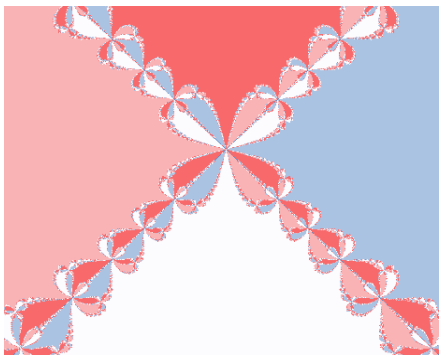
```
Sub Macro1( )
deltax = 4 / 300
deltay = 4 / 300
startx = -2
```

```
starty = 2

Cells.Select

Selection.ColumnWidth = 0.1

Selection.RowHeight = 1

   Range("A1").Select

For i = 0 To 300

For j = 0 To 300

x = startx + i * deltax

y = starty - j * deltay

  For k = 0 To 128

xn = x

qy = (x ^ 2 - y ^ 2) ^ 2 + 4 * x ^ 2 * y ^ 2 + 0.0000001

   x = (0.666667) * x + 0.333 * (x ^ 2 - y ^ 2) / qy

   y = 0.666667 * y - 0.6666667 * xn * y / qy

      If ((x + 0.5) ^ 2 + (y + 0.866025) ^ 2 <= 0.0001) Then

Selection.Cells.Offset(i, j).Interior.Color = RGB(255, Int(50 * Rnd()), Int(50 * Rnd()))

         Exit For

         End If

        If ((x - 1) ^ 2 + y * y <= 0.0001) Then

Selection.Cells.Offset(i, j).Interior.Color = RGB(Int(50 * Rnd()), 255, Int(50 * Rnd()))

         Exit For

      End If

            If ((x + 0.5) ^ 2 + (y - 0.866025) ^ 2 <= 0.0001) Then

Selection.Cells.Offset(i,j).Interior.Color= RGB(Int(50 * Rnd()), Int(50 * Rnd()), 255)

         Exit For

      End If

   Next k

 Next j

Next i

End Sub
```

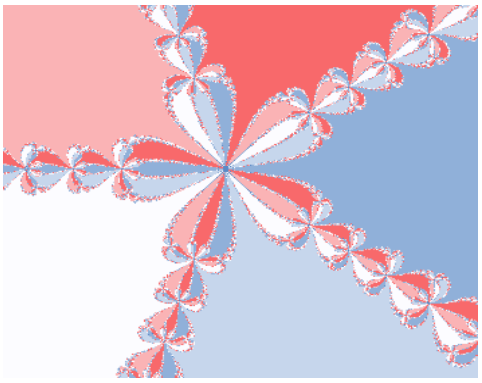
Fig:11 for n=3 is drawn with VBA, Others are drawn without VBA programming.



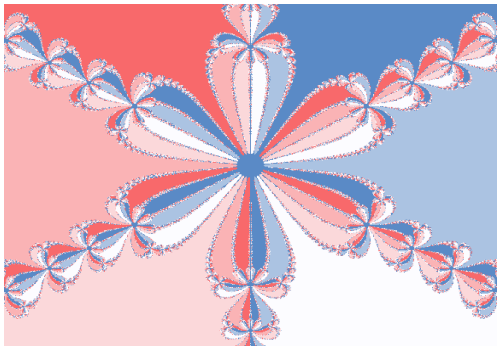
(a)



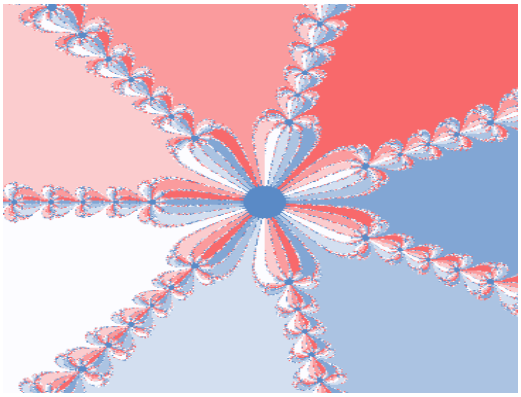
(b)



(c)



(d)



(e)

**Fig:11. Newton Fractal for n=3,4,5,6,7**

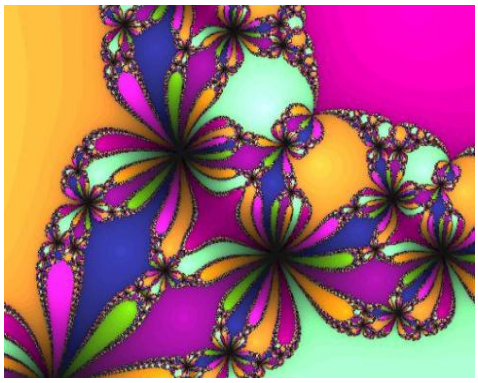
The excel sheets and snapshots are available at cen.amritafoss.org

With Programming, one will be able to draw complex Newton fractals like the ones shown in Figure 12 by taking different polynomials and color schemes.

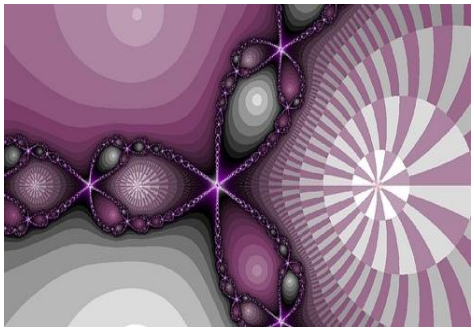
As pointed out in [1], the investigations of fractal and chaos is a marvelous topic for secondary school and for the university. When asked our post graduate students in Engineering to implement all kinds of fractals first in spreadsheet and then in Matlab and it is found that many students went on exploring different kinds of fractal without any further input. Some students even went on to read the book - Indra's Pearls: The Vision of Felix Klein [8] which explore the beautiful connection between group theory and geometry.

The output of fractal implementation shows the need for discovery approach to learning and the incorporation of technology to learn certain types mathematics which otherwise is impossible. One such area is discrete dynamical systems with applications to modeling the weather, the central nervous systems, and the stock market.

One of the aims in teaching mathematics should indeed be to "give the young a feeling for the beauty and eloquence of mathematics and its profound relationship with the real world", and it is difficult to see how the mathematics teacher can ignore an aspect of mathematics that is accessible with quite elementary mathematical ideas: fractal and chaos.



(a)



(b)

**Fig:12 Complex Newton Fractals**

## 4. SUMMARY

In this paper it is shown how Newton fractal can be drawn in Excel. What-If analysis tool provided in spreadsheet package is a powerful tool that can be used for variety of applications. Repetitive computation involving two variables can be easily done using What-If analysis tool. The approach is intuitive: show one example of computation, the rest of the computation is done by the What-If analysis tool. A variety of Newton Fractal can be produced in spreadsheet without programming. With programming one get more control over choice of color palette and also computation become extremely faster.

Fractals open the door to experimental mathematics- a new mathematics in which one need computer to explore the innumerable possibilities and it strengthen our intuition to create new mathematics.

## 5. REFERENCES

[1] Freitas, J.O.,Ramos, S., "*Computer Experiments with Newton's Method"*, 2003

[2] http://hal.inria.fr/docs/00/05/43/27/PDF/co37th2.pdf. Accessed 5 August 2012

[3] Tatham, S., "*Fractals derived from Newton-Raphson*". http://www.chiark.greenend.org.uk/ ~sgtatham/ newton/. Accessed 5 August 2012

[4] Bourke, P., **"***Gallery of fractals created using the Newton Raphson method***"**, http://paulbourke.net/fractals/newtonraphson/ (Accessed 5 August 2012)

[5] Barnsley,M.F., Devaney,R.L., Mandelbrot,B.B., Peitgen,H.O., Saupe,D., Voss,R.F., Heinz-Otto Peitgen, DietmarSaupe, "*The science of Fractal Images*", Springer-Verlag, 1988

[6] Barnsley,M., "*Fractals Everywhere*", Academic Press Inc, 1988

[7] Falconer,K., "*Fractal Geometry: Mathematical Foundations and Applications*",Wiley, 2003

[8] Peitgen,H., Juergens,H., and Saupe,D., "*Fractals for the Classroom",*Springer-Verlag, New York, 1992

[9] Mumford,D., Caroline Series David Wright , "*Indra's Pearls: The Vision of Felix Klein"* , Cambridge University Press, 2002

[10] Aravind, H., Rajgopal, C. and Soman,K.P., "*A Simple Approach to Clustering in Excel*", *International Journal of Computer Applications* 11(7):19–25, December 2010.