Nigerian Journal of Technology (NIJOTECH)
Vol. 36, No. 1, January 2017, pp. **155 – 162**

Print ISSN: 0331-8443, Electronic ISSN: 2467-8821
www.nijotech.com
http://dx.doi.org/10.4314/njt.v36i1.20

AUTOMATIC GENERATION OF ROOT LOCUS PLOTS FOR LINEAR TIME INVARIANT SYSTEMS

N. Durutoye[1,*] and O. Ogunbiyi [2]

[1] PEA DEPARTMENT, NIGERIA LIQUIDLY NATURAL GAS, BONY ISLAND, RIVERS STATE. NIGERIA.

[2] ELECTRICAL AND COMPUTER ENGINEERING DEPT, KWARA STATE UNIVERSITY, MALETE, KWARA STATE. NIGERIA.

E-mail addresses: [1] *durutonye.nathaniel@yahoo.com,* [2] *olalekan.ogunbiyi@kwasu.edu.ng*

ABSTRACT

Design and analysis of control systems often become difficult due to the complexity of the system model and the design techniques involved. This paper presents the development of a Tools Box in Microsoft Excel for control engineer that uses root locus as a time domain technique for system design and analysis. The Tool Box can also serve as a computer-aided graphical analytical tool for trainers. The work was done in two phases: the first phase is the development of a programmable algorithms for root locus using the angle condition and bisection method while the second phase is the implementation of the developed algorithms. The implementation was done using Microsoft Excel (R) Visual Basic Application (VBA). Results of simulations for different systems show the potential of the Tool Box as an alternative for other software package and the ease of using it on the readily available Microsoft Excel environment.

Keywords: *Asymptotes, Poles, Root Locus, Singularity Point, Zeros*

1. INTRODUCTION

The evolution of Control Engineering practice has propelled the performance of any design system to a reasonable satisfaction which can always be improve on continually basis. Since control system design is invariably an iterative process, the computer, combined with reliable software remove the tedium of performing repetitive calculation and/or analysis [1, 2].

The relationship between computer application and educational sector has introduced a mechanism that facilitates high level of understanding to learning. It brings theoretical analysis of a particular topic to practical reality.

An advancement in the study of control theory is the development of method for graphically extracting the system root in a continuous manner as some parameters varies. This method referred to as Root locus techniques and was developed by Evans in 1948 [3]. It has become well-known and it has survived the innovation in control theory, it has equally been applied to classical formulation and the more recent state variable approaches. Root locus techniques is referred to as a method that extract all root of characteristic equation using a set of rules derived from general property of the transfer functions. It helps the designer to predict the effects on the location of the closed-loop poles of varying the gain value and adding open-loop poles or open-loop zeros. In addition to its capability to describe the effect of varying gain upon percent overshoot, settling time, and peak time, its real power is its ability to solve problems with higher order systems. It provides a graphical representation of a system's stability such that the designer can clearly see ranges of stability [1, 4].

The performance of a feedback control system depends greatly on the location of the root of the characteristics equation in the s-plane. Root locus technique investigates the trajectories of the root of the closed loop system characteristics equation (root loci) as a particular parameter is being varied. The root locus plot is a powerful and reliable tool in the analysis of feedback control systems as presented by various authors [3, 5, 6].

The plot may become easier to sketch once the fundamentals are well understood but it is computationally challenging to generate automatically. Using Microsoft Excel® VBA (Visual Basic Application) to solve root locus in a common application that is readily available to every student and instructor (teacher) is highly essential [7, 8], it is not expensive in comparison with MATLAB or any other control application software. An Excel® VBA was developed in this work to generate the root locus; this is one of the time domain analysis through which a system design and performance can be optimized [3].

*** Corresponding author, tel: +234 – 703 – 030 – 5327**

In designing any system, time always happen to be a major constraint, therefore the use of complex graphical and mathematical tools which required long hours of manual implementation may not be appropriate. Manual calculations for higher order system such as four (4) and above may be quite unpleasant to solve [5, 9].

Another inspiration evolves from experience in that: due to the nature of control engineering option in electrical field, students are always running away from this option owing to the fact that its theorems and principles looked somehow abstract in nature. As a result, this computer aided application in Excel® VBA will help the students to perform a lot of control experiments on easily accessed Microsoft office package.

The problem addressed in this paper is the development of an Excel® VBA module that will facilitate the generation of the Root Locus plots for teaching and design purposes. The objectives are to; develop a suitable iterative algorithm for generating Root Loci and realize the resulting algorithm using the commonly available Excel® VBA software. The extent of the work done does not address either the case of computer controlled systems or systems with time delay. In other words, only strictly linear time invariant systems were considered. Different algorithms were used in the past [5, 6, 9] but some become intractable in some situation. This work uses a search algorithm technique, which can be embedded in most software.

2. SYSTEM DESIGN

The objective of this research is the development and implementation of a computer program for the automatic generation of root loci using Microsoft Excel® VBA (Visual Basic Application). The algorithm employed for this purpose, the organization and documentation of the resultant program are discussed in this section.

Given a feedback control system shown in Figure 1, the closed-loop transfer function $T_{(s)}$ of the system is given as:

$$T_{(s)} = \frac{C_{(s)}}{U_{(s)}} = \frac{G_{(s)}}{1 + G_{(s)}H_{(s)}} \quad (1)$$

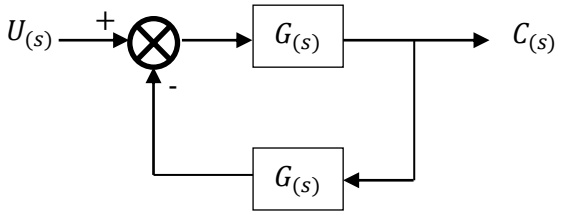

Figure 1 Feedback system with open loop transfer function $G_{(s)}H_{(s)}$

The characteristic equation Q(s) for this closed-loop system is obtained by setting the denominator of the right-hand side of equation 1 to zero.

$$Q_{(s)} = 1 + G_{(s)}H_{(s)} = 0 \quad (2)$$

Considering a variable K in the open loop transfer function $G_{(s)}H_{(s)}$ such that.

$$G_{(s)}H_{(s)} = \frac{KA_{(s)}}{B_{(s)}} \quad (3)$$

Where, $A_{(s)}$ are the zeros and $B_{(s)}$ are poles of the open loop system and are polynomials of orders m and n respectively, equation 2 can be rewritten as

$$1 + \frac{KA_{(s)}}{B_{(s)}} = 0 \quad (4a)$$

$$B_{(s)} + KA_{(s)} = 0 \quad (4b)$$

The characteristic equation of a typical linear time invariant system can be expressed as shown in equation (5).

$$1 + KF_{(s)} = 1 + K\frac{\prod_{j=1}^{m}(s - z_j)}{s^q \prod_{i=1}^{n}(s - p_i)} = 0 \quad (5)$$

In (5), K is the root locus gain; $F_{(s)}$ is the open loop transfer function $z_j, j = 1, 2, 3 \ldots m$, is the set of finite open loop zeros and $p_i, i = 1, 2, 3 \ldots n$, is the set of finite open loop poles. Available algorithms for sketching this root locus can be categorized as follows:

i. ***Direct Methods:*** These are the algorithms in which various numerical schemes are used to determine the roots of the polynomial as K is varied in some prescribed manner. Since this algorithm is limited to polynomials, they cannot handle systems with dead time such as time delay systems.

ii ***Area Search Algorithms:*** These classes of algorithms generate the root loci by grid-search techniques over a specified area of the complex frequency - plane (s-plane). The grid search method, in effect is used to determine the points sanctifying the necessary and sufficient conditions for root locus points. Any of the angle or magnitude condition equation can serve as the search criterion.

iii. ***Branch Following Algorithm:*** This method was first presented by R.H. Ash and G.R. Ash in 1968. The algorithm in this case often presume "a prior" knowledge of a point on the locus of interest and use the geometric and trigonometric asymptotes properties of the locus at the given point to predict a new point on the locus. An algorithm employed in this research is based on Branch Following Algorithm.

2.1 Sketching the Root Locus

Plot of root locus of a system in Excel VBA® was carried out by following the algorithm below:

(i) Determination of the number of poles (n)and the number of zeros (m) of $G_{(s)}H_{(s)}$.

(ii) Determination of the location of the poles and the number of zeros of $G_{(s)}H_{(s)}$.

(iii) Determination of the number of branches. The root locus will have a total of branches equal to the number of poles in the open loop transfer function.

(iv) Loci of the real axis: The sections of the root locus on the real axis of the complex plane are determined by counting the total number of finite poles and zeros of $G_{(s)}H_{(s)}$. to the right of the points in question. For values of $K > 0$, points of the root locus on the real axis lie to the left of an odd number of finite poles and zeros.

(v) Determination of the asymptotes of the root loci. For large distances from the origin in the s-plane, the branches of a root locus approach a set of asymptotes. These asymptotes emanate from a point in the complex plane on the real axis called the center of asymptotes σ_c given by

$$\sigma_c = -\frac{\sum_{i=1}^{n} p_i - \sum_{j=1}^{m} z_j}{n-m} \quad (6)$$

The angle of the asymptotes are determined as

$$\theta = \pm \frac{180^0(2k+1)\pi}{n-m}, \quad k = 0,1,2,3,\ldots \quad (7)$$

Where p_i are the poles, z_j are the zeros, n is the number of poles and m the number of zeros.

(vi) Break-away and Break-in Points: The conventional breakaway/break-in points are defined as the points on the real-axis at which a root locus branches leave (break away from) or enter (break into) the real axis. The location of the breakaway point can be determined by solving the following equation for σ_b.

$$\sum_{i=1}^{n} \frac{1}{(\sigma_b + p_i)} = \sum_{j=1}^{m} \frac{1}{(\sigma_b + z_j)} \quad (8)$$

A breakaway or break -in point exists if and only if two singularities odd and even are of the same type or if such a pair is complemented by a branch emanating from its opposite kind from infinity. This clue paves a way to logically program and greatly simplify the algorithm.

A logical procedure was incorporated into the algorithm such that for every pair of singularities that exist a break point, an $\varepsilon = 0.02$ is assigned to y-axis value and the procedure iterate horizontally. If it converges, the point on the x-axis at which it converges is taken as break-away or break-in point, otherwise there is no break-away or break-in point at that segment of the root locus.

(vii) Departure and Arrival Angles:

The departure angle of the root locus from a complex pole is:

$$\theta_D = 180^0 + \angle GH \quad (9a)$$

and the arrival angle is:

$$\theta_D = 180^0 - \angle GH \quad (9b)$$

(viii) Plotting and calibration of the root locus using the angle condition and bisection method.

(a) Angle Condition: In this work, the technique used for plotting the locus was based on the angle condition expressed from the characteristic equation (5).From the concepts of complex variables, the angle condition can be expressed as equation (10).

$$\sum Arg(s - z_m) + Arg(s - p_n) - \frac{q\pi}{2} - (2r+1)\pi = 0 \quad (10)$$

Where $r = 0, 1, 2, 3 \ldots (n - m - 1)$ and q is the system type order.

The algorithm works in such a way that points in the s-plane must be determined such that they satisfy the angle condition. For an arbitrary point s in the complex plane, let $Arg\{F_{(s)}\}$be an angle function defined relative to the gain normalized open loop transfer function such:

$$\sum Arg\{F_{(s)}\} = (2r+1)\pi + \frac{q\pi}{2} \quad (11)$$

$$Arg\{F_{(s)}\} = \sum_{m=1}^{M} Arg(s - z_m) + \sum_{n=1}^{N} Arg(s - p_n) \quad (12)$$

Let exploratory point (test point)$s = x + jy$

$$F_{(s)} = F(x + jy) \quad (13)$$

Therefore, equation 12can be rewritten as;

$$Arg\{F_{(s)}\} = \sum_{m=1}^{M} tan^{-1}\left(\frac{y - y_m}{x - x_m}\right) - \sum_{n=1}^{N} tan^{-1}\left(\frac{y - y_n}{x - x_n}\right) = \theta \quad (14)$$

Where $z_m = x_m + jy_m$, $for\ m = 1,2,3, M$ is the set of finite open loop zeros.

$p_m = x_n + jy_n$ $for\ n = 1,2,3, \cdots N$ is the set of finite open loop poles.

Hence, a pointS_0 lies on the root locus if and only if it satisfies the angle condition (equation 10).

$$F_{(s_0)} = \left\{\sum_{m=1}^{M} tan^{-1}\left(\frac{y - y_m}{x - x_m}\right) - \sum_{n=1}^{N} tan^{-1}\left(\frac{y - y_n}{x - x_n}\right)\right\} - (2r+1)\pi = \theta \quad (15)$$

Equation (15) indicates that the calculation of legitimate root loci points can be achieved by finding the zeros of the angle function - a transcendental equation.

Many techniques may be used but the need to calculate derivatives could limit the complexity of systems that can be solved, so a derivative-free method is usually preferred. As a result, in this work, the bisection method was used.

A simple scheme for achieving the iterative process can start with some θ_0 and then determine the improvements,θ_0 on the initial guess using the "bisection method". In addition to the ability to determine suitable converging iterates, the bisection method requires that at start-up, two values (singularities) of the variable that bracket the real value must be known. This implies that at every stage of the computation, two points s_1 and s_2 must be available such that$F_{(s_1)} > 0$ and $F_{(s_2)} < 0$.

bisection method for the angle loci

Let x be the real and y be the imaginary part of function$F_{(s)}$define for the real part of a point on the locus, $l_o \leq x \leq h_i$ *or* $l_o \leq x \leq h_i$. " l_o " and " h_i " are interval search area which can be in coordinate of x or $y$and are initialize by asymptotic line. In this interval, if$F_{(s)} = (2r+1)\pi + \frac{q\pi}{2}$, correspondingly an exploratory point that make $F(x+jy)$ to equals $(2r+1)\pi + \frac{q\pi}{2}$is known as a closed loop pole$F_{(s)}$.

In order to implement the bisection method, a simple test is needed, to see if function$F(l_o, y)$, has an argument (angle) greater thanπ *or* $(2r+1)\pi + \frac{q\pi}{2}$and if function $F(h_i, y)$, has an argument less than $|180|$ or$|n*180|$then that $x$the real part of the root (locus) exist in that interval.

The process involved in the search and check can be described by Figure 1, such that:

$$\angle\{G_{(s)}H_{(s)} = \sum_{n=1}^{N} \angle(s_E - p_n) - \sum_{m=1}^{M} \angle(s_E - p_m) \quad (16)$$

$$F(x+jy) = (2r+1)\pi + \frac{q\pi}{2} \quad (17)$$

$$F(h_i + jy) = \theta_1 + \theta_2 + \theta_3 \quad (18)$$

$$F(h_o + jy) = \theta_4 + \theta_5 + \theta_6 \quad (19)$$

$$m = \frac{1}{2}(l_o + h_i) \quad (20)$$

Surely the contribution of the arguments depends on the part of the s-plane being considered. If one is iterating horizontally, there will usually be two bounds depending on how close to the real singularities and the asymptotes.

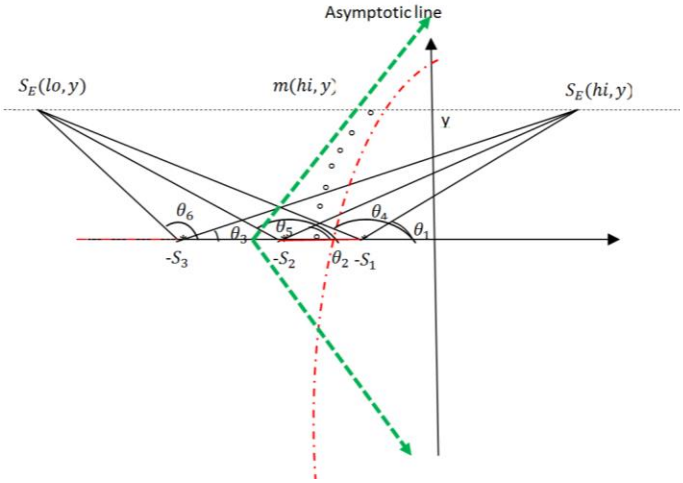

Figure 1: Angle measurements from open loop poles and open loop zero to test point S_E through bisectional method.

A solution would simply use the asymptote for the branch as one boundary and the other can then be determined by actually trying small steps away from it till the sign changes or else intelligently use the bounds established by the singularities that originate the branch. Once the interval has been established it is trivial to determine the final best value. Consequently, an important function is the subroutine which works for each singularity and determines the argument of $arg\{s - w_l\}$ where w_l is any of the $N+M+q$ singularities that define the open loop transfer function $G(s)H(s)$.

$$F(h_i + jy) = ? \quad (21)$$

The first step in the bisection method involves dividing (bisecting) the interval $l_o \leq x \leq h_i$ into two subintervals $l_o < x < m$ and $l_o < x < h_i$ where $m = \frac{1}{2}(l_o + h_i)$ the subinterval to be considered next is obtained by replacing l_o by m if $F(m+jy) > |\pi|$or$(2h+1)\pi + \frac{q\pi}{2}$ because in this case $F(m+jy) > |\pi|$ or $(2h+1)\pi + \frac{q\pi}{2}$ but less than $F(l_o + jy)$. Conversely, if$F(m+jy) < |n\pi|$ then subinterval to be considered is obtained by replacing h_i by m because in this case $F(m+jy) < |\pi|$ or $(2h+1)\pi + \frac{q\pi}{2}$ but greater than$F(m+jy)$, therefore, the next subinterval will exist as $l_o < x < m$ and so this interval must contain a root(x, y). Note that $l_o$and h_i are taking to be boundary under which root of the function are bracketed and is determine by asymptotic line. The task of finding the root has now been refined from considering the interval $l_o \leq x \leq m$ and replaced by the task of finding the unknown coordinate part of the root in an interval half the size until a true value is found. Once x value is found, $y$value will be step up or step down by a constant value base on the form that characteristic equation takes.

2.2 Root Locus Performance Parameters

i. Gain Margin: This is defined as the factor by which the design value of K must by multiplied before the closed loop system becomes unstable, i.e.

$$M = \frac{K_x}{K_D} \quad (22)$$

Where K_x is value of system parameter K on the imaginary axis of s-plane and $K_D$is the design value of system parameter K.

ii. Phase Margin: This is the sum of 180^o and the phase angle of the open loop transfer function at the point where the magnitude of the open loop transfer function is equal to unity given by;

$$PM = \emptyset_{PM} = 180^0 + \angle\, GH_{(jw)} \quad (23)$$

3. SOFTWARE DESIGN

A modular approach was employed at the design stage of this control toolbox, the first step taken was to decompose the whole program into a mutually exclusive set of procedures (subroutines or functions) such that the combined total of all with the main body of the program can fully implement the process described in the outline shown below. Consequently, each procedure was then developed separately before final integration to

form a whole module as shown in the modular plan of the whole system. Each Procedure in this case is referred to either as a subroutine or a function in an Excel(R) VBA application.

Excel VBA Root Locus Toolbox is a user friendly design toolbox; it accepts data through some cells and display information in some, the execution of a program is activated through a "Compute root-locus". The interface can be categorized into two; input and output cells. In figure 2, point "A" is initiated with spin button which inputted numbers of both poles and zeros in a transfer function of a given system, the number of singularities in transfer function can be increment or decrement by clicking on these buttons. Point "B" comprises of three rows, each three rows for both pole and zero, each rows is caption with "Real part =, imaginary part =, and symbol", real part are the real coefficient of the singularity, imaginary part are the imaginary coefficient of the singularity while symbol are initialized with "*p*" for poles and "*z*" for zeros. Point "*C*" are used to initialize the boundary of generated chart, that is, extend of the size of the chart, this point comprises of four rows and value must be inputted there. Point "*D*" is initiated with command button and mark the execution of the program. Point "*E*" only display the singularities inputted in point "*B*" in descending order. Point "*F*" is a series of cells that display the generated root locus. Point "*G*" is a chat that is plotted with the generated roots.

4. SIMULATIONS AND RESULTS

Implementation of the algorithm was carried out using Microsoft Excel® VBA. It has a number of control buttons that facilitate the adjustment of the system parameters. The numbers of poles and zeros can be entered directly or by means of two spin buttons. System type order can similarly be adjusted or entered using a corresponding spin button. These completely determine the character of the open loop system. Initiation of the root-locus calculation is carried out by clicking on the command button designated as "Compute root-locus". The final realization of the root-locus tool has an opening window as shown in Figure 2.

4.1 Tests of the Root-Locus Generator

The ability to generate the root-loci for a given system depends on the structure and relative placement of the poles and zeros of the system. Three broad categories can be recognized while a fourth is a combination of the other types. The first category includes different types of systems with all poles and no zeros. The next class has a combination of poles and zeros. This has two sub-classes where the difference between the number of poles and zeros are either 0, 1 and greater than 1. In the first two sub-classes there may be closed loci depending on the location of the singularities on the real axis. These are tested in the following experiments. The computation time is observed to depend on the following parameter: The size of the system: that is the number of singularities (poles and zeros) of the system; computation time are more for complex transfer function. The step size, "∂" (the distance between each point): the bisection iteration converges faster for larger values of ∂ and this reduces number of point generated.

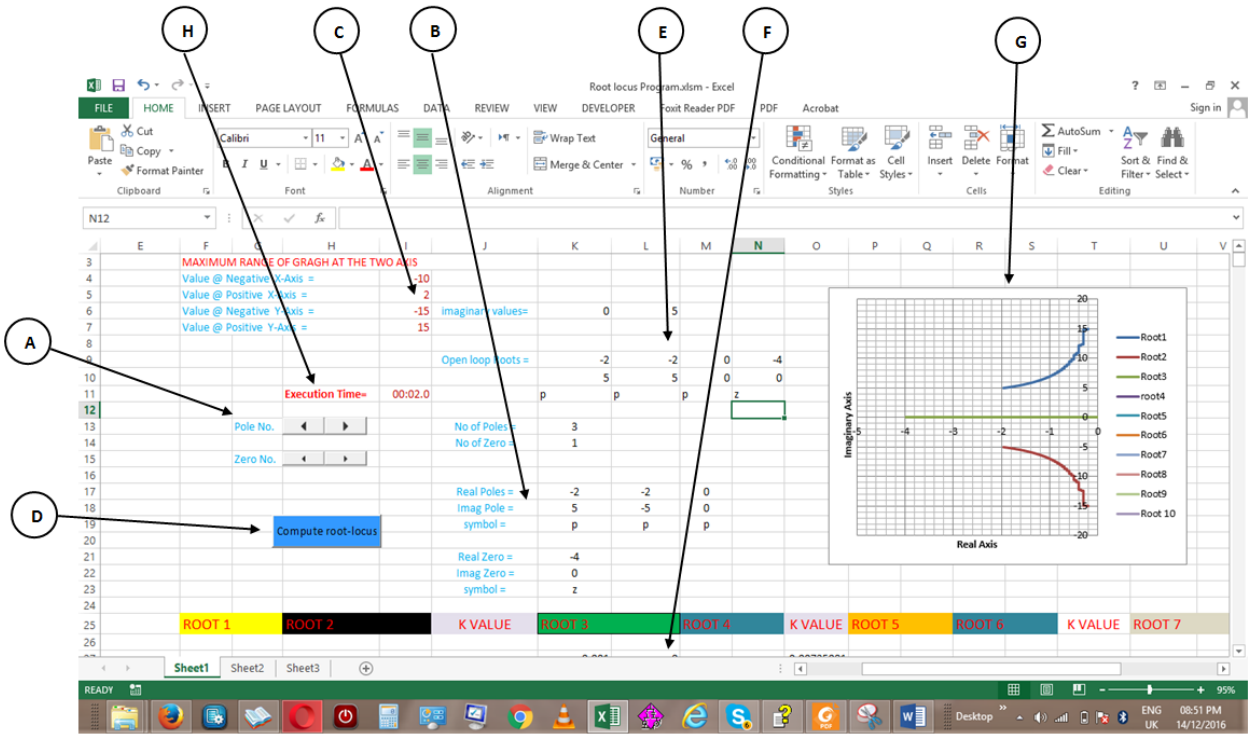

Figure 2: GUI Interface on Microsoft Excel

The number of branches: from the tested transfer function the number of branches is seen to be a function of time, the high the number of branches the high the execution time.

The root-locus tool determines its different branches such that whenever a transfer function is defined and the command button clicked, it first determines all the branches on the real axis where they exist, then vectoring to determine either break-in or break-away points as the case may be between the odd and even singularities on the real axis. The break branches are then determined using the asymptotes as bounds for trials and the roots are extracted progressively for increasing values of the imaginary part of the roots.

The simulations presented here reflect a progressive increase in complexity where first all the singularities are poles, then a progression of zeros are added to the transfer function with the location of the zeros being modified to change from the trivial when the successive odd and even singularities are of opposite type (trivially no break-away or break-in points) to more complex cases where there may be both break-away and break-in points. The simulations for different systems are as presented in Figures 3 to 12.

The normalized transfer function "$G_{(s)2}$" has two poles and one zero. The iteration started by pairing the singularities into odd and even number from right to left for real singularities. It checked if both odd and even singularities are of the same type, in this case, the first set of odd and even singularities are of different type (the first singularity pole p = -1 and the second singularity zero z = -2). It is certain that there will not be a break branch and path of locus along the real axis cannot exceed the range of paired singularities (odd and even singularities) therefore, a small decrement say, step size "∂ " = 0.01 is deducted from most positive singularity until it reaches the second singularity, at each deduction, test point angle is checked to confirm if it equal to corresponding angle conduction (see equation 3 and 4), if it equals, the corresponding test point is taken as root for that point. The first segment of the root terminated immediately step size "∂" decrement equal to the second root of that paired singularity. The iteration proceeds to the next paired root but in this case, it is only one root that is left, since only remaining root is real root with no imaginary, it follows the same iteration process as observe in first paired root. In this case the search root tends to infinity and intentionally truncated once "∂" decrement equal chart boundary along negative x-axis.

The normalized transfer function "$G_{(s)3}$" has two poles and one zero like "$G_{(s)2}$" "but the position of the zero in this case is different , the iteration started by pairing the singularities, it checks if paired singularities are of the same type, in this case the singularities are of the same type (the first is pole p = -1 and the second is pole p = -3), it is certain that there will be a break branch and path of locus along the real axis cannot exceed the range of paired singularities therefore, a small decrement say, step size "∂ " = 0.01 were deducted from most positive singularity until it reaches the second singularity, at each decremented step, test point angle is compare with corresponding angle conduction, if the compared test point is valid, the corresponding test point is taking as root for that point. The breaking point is determine by changing a step size "∂ " in real axis to imaginary axis from y = 0 to a step size "∂ " = 0.01 for a start and iterate horizontally for every increment. The program has a logical operation to change or determine path of search either by vertical or horizontal iteration especially for function that has sphere topology.

The normalized transfer function "$G_{(s)2}$" has two poles and one zero. The iteration started by pairing the singularities into odd and even number from right to left for real singularities. It checked if both odd and even singularities are of the same type, in this case, the first set of odd and even singularities are of different type (the first singularity pole p = -1 and the second singularity zero z = -2). The normalized transfer function "$G_{(s)2}$" has two poles and one zero. The iteration started by pairing the singularities into odd and even number from right to left for real singularities. It checked if both odd and even singularities are of the same type, in this case, the first set of odd and even singularities are of different type (the first singularity pole p = -1 and the second singularity zero z = -2).

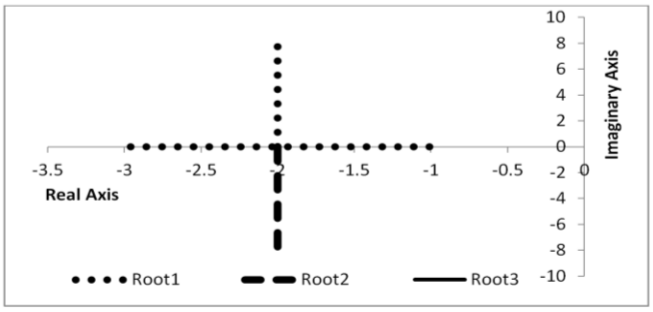

Figure 3: Root-locus plot for $G_{(s)1} = \frac{1}{(s+1)(s+3)}$

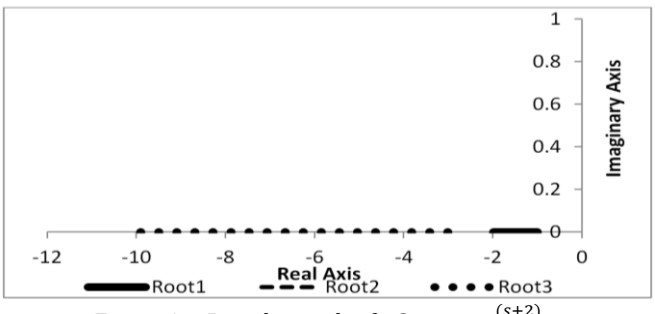

Figure 4: Root-locus plot for $G_{(s)2} = \frac{(s+2)}{(s+1)(s+3)}$

It is certain that there will not be a break branch and path of locus along the real axis cannot exceed the range of paired singularities (odd and even singularities) therefore, a small decrement say, step size "∂ " = 0.01 is deducted from most positive singularity until it reaches the second singularity, at each deduction, test point angle is checked to confirm if it equal to corresponding angle conduction (see equation 3 and 4), if it equals, the corresponding test point is taken as root for that point. The first segment of the root terminated immediately step size "∂" decrement equal to the second root of that paired singularity. The iteration proceeds to the next paired root but in this case, it is only one root that is left, since only remaining root is real root with no imaginary, it follows the same iteration process as observe in first paired root. In this case the search root tends to infinity and intentionally truncated once "∂" decrement equal chart boundary along negative x-axis.

The normalized transfer function "$G_{(s)3}$" has two poles and one zero like "$G_{(s)2}$" "but the position of the zero in this case is different , the iteration started by pairing the singularities, it checks if paired singularities are of the same type, in this case the singularities are of the same type (the first is pole p = -1 and the second is pole p = -3), it is certain that there will be a break branch and path of locus along the real axis cannot exceed the range of paired singularities therefore, a small decrement say, step size "∂ " = 0.01 were deducted from most positive singularity until it reaches the second singularity, at each decremented step, test point angle is compare with corresponding angle conduction, if the compared test point is valid, the corresponding test point is taking as root for that point. The breaking point is determine by changing a step size "∂ " in real axis to imaginary axis from y = 0 to a step size "∂ " = 0.01 for a start and iterate horizontally for every increment. The program has a logical operation to change or determine path of search either by vertical or horizontal iteration especially for function that has sphere topology.

5. CONCLUSION

Excel VBA Root-locus generator was tested using systems of different transfer function. Clearly the degree of complexity depends on the structure of the transfer function. For example, those with no breakaway points are generally simpler to generate then close loop poles than those with breakaway points. While the systems tried in this work are not exhaustive of all possibilities, it can be seen that the procedure can solve a reasonable number of systems that one may encounter in real life. This is a very welcome result since it is now possible for a designer, instructor or student to explore the root-locus characteristics of many linear time invariant systems using nothing more proprietary than the ubiquitous EXCEL with its VBA environment.

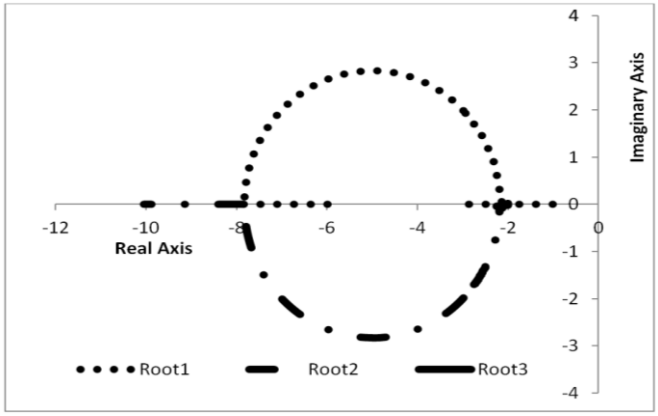

Figure 5: Root-locus plot for $G_{(s)3} = \frac{(s+5)}{(s+1)(s+3)}$

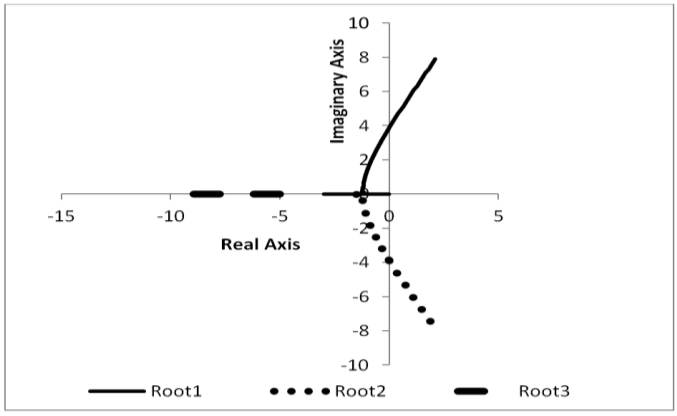

Figure 6: Root-locus plot for $G_{(s)4} = \frac{1}{s(s+3)(s+5)}$

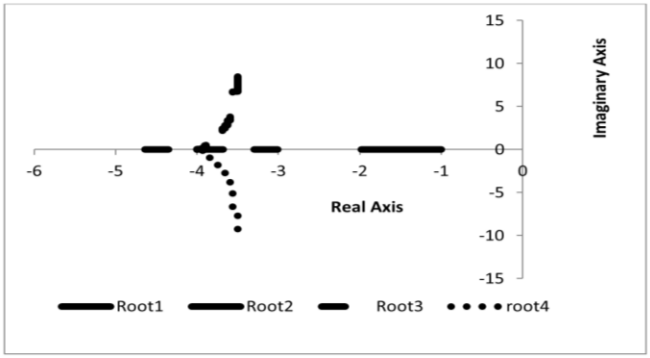

Figure 7: Root-locus plot for $G_{(s)5} = \frac{(s+2)}{(s+1)(s+3)(s+5)}$

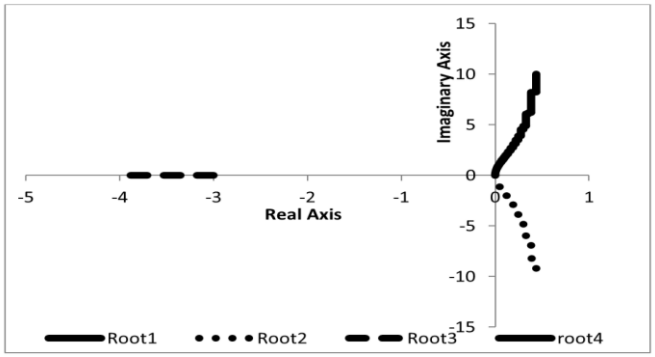

Figure 8: Root-locus plot for $G_{(s)6} = \frac{(s+4)}{s^2(s+3)}$

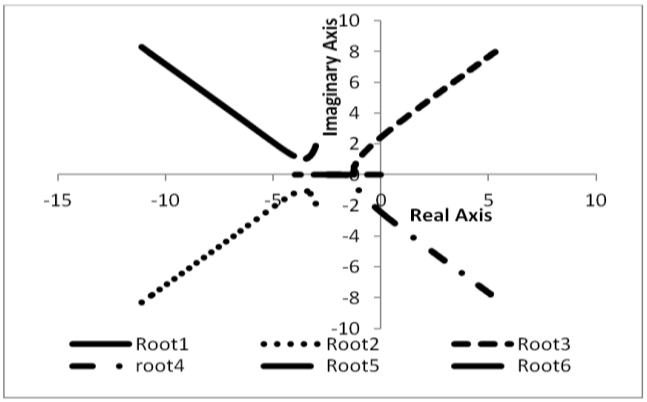

Figure 9: Root-locus plot for $G_{(s)7} = \frac{1}{s(s+3+2j)(s+3-2j)(s+5)}$

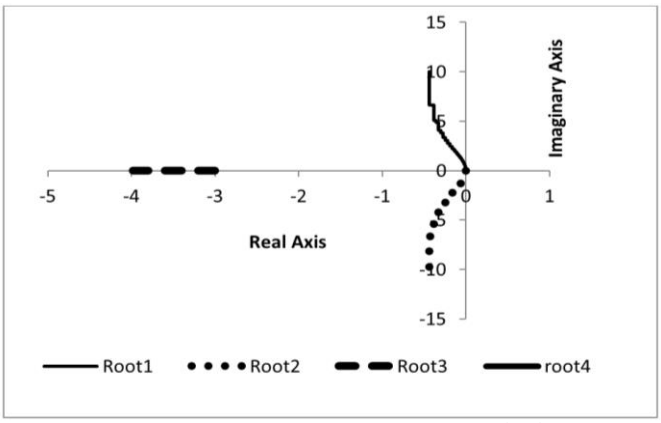

Figure 10: Root-locus plot for $G_{(s)8} = \frac{(s+3)}{s^2(s+4)}$

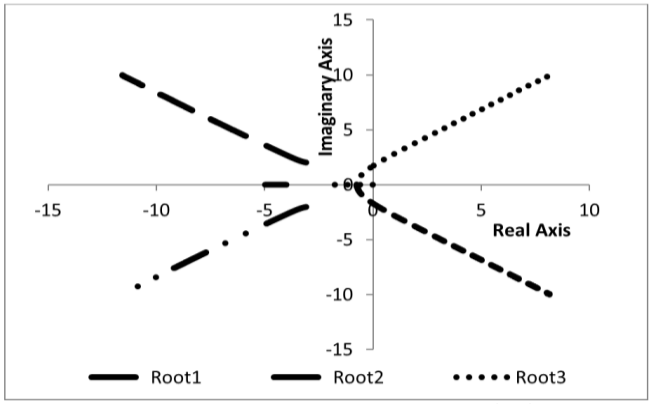

Figure 11: Root-locus plot for $G_{(s)9} = \frac{(s+5)}{s(s+2)(s+4)(s+3+2j)(s+3-2j)}$

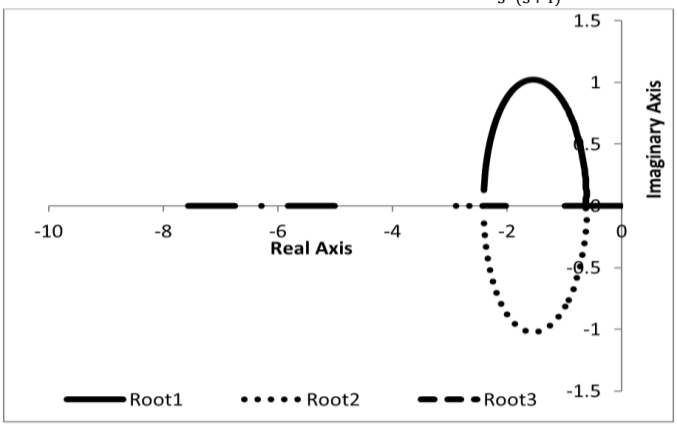

Figure 12: Root-locus plot for $G_{(s)10} = \frac{(s+2)(s+3)}{s(s+1)(s+5)}$

Since this was essentially the objective of this work, it can be concluded that it has been successfully resolved. It is possible to generate root-locus plots for a sufficiently wide range of system types and use this for teaching, laboratory and design purposes.

6 RECOMMENDATIONS

The implementation presented here depends on search process which is not general but attempts to identify all possible types of loci for specific cases. This is a limitation which should be addressed by a subsequent researcher so that the resulting algorithm can be more compact and efficient. It is recommended that the algorithm be made more efficient by modifying its branching logic and using a more compact approach.

7. REFERENCES

[1] K, Ogata, *Modern Control Engineering*, 3rd ed., Prentice Hall, Upper –Saddle River, NJ, 1997.

[2] A. Grace, A. J. Lamb, J. N. little, and C.M. Thompson, *Control System Toolbox for use with MATLAB, User guide*, The Math works, Inc , Natick, Mass, 1992.

[3] R.C. Dorf and R.H. Bishop, *Modern Control Systems*, 11th ed., Prentice Hall, 2008.

[4] S. N. Norman, *Control System Engineering*, 3rd ed., John Wiley & Sons, New York, USA, 2000.

[5] J. F. Opadiji, *Development of Software for Graphical Analysis of Continuous-Time Control Systems, M.sc, Thesis*, University of Ilorin. 2003.

[6] J. J. Blakley, "*An expert system root locus plotter*", *INT J EL EN*, vol. 36, no. 4, pp.298-310. 1999.

[7] L. M. Robert, *An Introduction to VBA in Excel*, 2nd ed, Kellog School, Northwestern University, 2000.

[8] D. Birnbaum, *Microsoft Excel VBA Programming*, 2nd ed., Thomson Course Technology PTR, 2005.

[9] K. C. Okafor, *Automatic Generation and Plots of Root Loci*. B.sc, Thesis, Ahmadu Bello University, Zaria., 1981,

[10] J. Moscinski, and Z. Ogonowski, *Advance Control with MATLAB and SIMULAINK* , Ellis Horwood, Hemel Hempstead, UK, 1995.

[11] S.D. Conte and C. de Boor, *Elementary Numerical Analysis: An* Al*gorithmic Approach*, 3rd ed., McGraw-Hill, New York ,1981

[12] R. S. Burn, *Advance Control Engineering*, 1st ed., Butterworth-Heinemann, 2001.

[13] G. T. John, *Automatic Feedback Control System Synthesis*, McGraw-Hill book company, Inc, USA, 1955.

[14] P. N. Paraskevopoulos, *Modern control Engineering*, 1st ed., Marcel Dekker Inc, New York USA, 2002.

[15] F. Raven, *Automatic control Engineering*, Second edition, McGraw-Hill, New York, 1990.

[16] J. G. Walter and L. V. Thompson, *Modern Control Systems Analysis and Design*, John Wiley and Son Inc. USA, 1993.