

**PRECISE DETERMINATION OF THE**  
**MULTIPLE ROOTS OF**  
**HIGH ORDER POLYNOMIALS**  
**(2)**  
**THE CASCADE DIFFERENTIAL**  
**ANALYSIS METHOD.**

**Peter G.Bass.**

**Abstract**

This paper describes the Cascade Differential Analysis method for the precise determination of the multiple plus singular roots of polynomial equations of any order greater than three. Polynomials of order three or less are covered by other means. The method is applicable when any polynomial contains multiple plus singular roots in any configuration.

**1 Introduction.**

It is well known that if a polynomial of order  $n$  contains  $p$  multiple roots, the  $(p - 1)$ th differential will contain one of the multiple roots, plus  $(n - p - 1)$  other roots reflecting the dynamics of the original equation. Mostly these other roots will be singular, but in a few cases, can also contain a remaining multiple root, if the original equation contained more than one multiple root in equal quantities. The Cascade Differentials Analysis, (CDA), method described here, is an analytic, (non-iterative), method which will accurately determine the multiple plus singular roots, in any configuration, of polynomials of any order greater than three.

**2 Description of the Method.**

**2.1 The Cascade Differential Analysis Method.**

The method is extremely simple, requiring no special mathematical development. For a polynomial of order  $n$ , first take the  $(n - 3)$ th differential. This will be a 3rd order equation. Find the roots of this derivative and in turn divide them into the original polynomial for a zero remainder. If none of these trial roots yield a result, take the  $(n - 4)$ th differential and perform the same process. Continue this exercise until a trial root provides a zero remainder, and is therefore a multiple root of the original equation or, until all derivatives are exhausted. If a trial root yields a result, it is then divided out of the original equation, and the reduced polynomial subjected to the same procedure to look for additional multiple roots.

Should all the derivatives from the  $(n - 3)$ th to the first be exhausted, without finding a multiple root, then the original polynomial contained only singular roots which can be determined by any other method.

### **2.1.1 Implementation.**

Due to the nature and number of computations required by this method to analyse high order polynomials, it is more suitable to computer analysis rather than manual.

The method is implemented here via an addition to the BAIRSTOW2.XLS Excel spreadsheet process associated with, and described in [1] and [2]. The updated spreadsheet, BAIRSTOW3.XLS, plus an ancillary spreadsheet, EQHOLD.XLS, plus the Polynomial Construction spreadsheet, POLYNOMIALCONSTRUCTION.XLS, can be downloaded here,

### **BAIRSTOW3.ZIP.**

EQHOLD.XLS must be placed in the root directory, i.e. C:\.

### **2.1.2 Coverage.**

In BAIRSTOW3.XLS only the  $(n - 3)$ th derivative is first taken to look for multiple roots, because such roots applicable to the  $(n - 1)$ th and  $(n - 2)$ th derivatives are covered by the DDR Module. Thus the combination of multiple roots covered by the CDA method, for polynomials of order 4 to 10, is as per the following table.

Order	Combination	Order	Combination	Order	Combination
4	2+1+1	9	7+1+1	10	8+1+1
	3+1+1		6+2+1		7+2+1
5	2+2+1	9	6+1+1+1	10	7+1+1+1
	2+1+1+1		5+3+1		6+3+1
6	4+1+1	9	5+2+2	10	6+2+2
	3+2+1		5+2+1+1		6+2+1+1
6	3+1+1+1	9	5+1+1+1+1	10	6+1+1+1+1
	2+2+2		4+4+1		5+4+1
6	2+2+1+1	9	4+3+2	10	5+3+2
	2+1+1+1+1		4+3+1+1		5+3+1+1
7	5+1+1	9	4+2+2+1	10	5+2+2+1
	4+2+1		4+2+1+1+1		5+2+1+1+1
7	4+1+1+1	9	4+1+1+1+1+1	10	5+1+1+1+1+1
	3+3+1		3+3+3		4+4+2
7	3+2+2	9	3+3+2+1	10	4+4+1+1
	3+2+1+1		3+3+1+1+1		4+3+3
7	3+1+1+1+1	9	3+2+2+2	10	4+3+2+1
	2+2+2+1		3+2+2+1+1		4+3+1+1+1
7	2+2+1+1+1	9	3+2+1+1+1+1	10	4+2+2+2
	2+1+1+1+1+1		3+1+1+1+1+1+1		4+2+2+1+1
8	6+1+1	9	2+2+2+2+1	10	4+2+1+1+1+1
	5+2+1		2+2+2+1+1+1		4+1+1+1+1+1+1
8	5+1+1+1	9	2+2+1+1+1+1+1	10	3+3+3+1
	4+3+1		2+1+1+1+1+1+1+1		3+3+2+1+1
8	4+2+2	9		10	3+3+1+1+1+1
	4+2+1+1				3+2+2+2+1
8	4+1+1+1+1	9		10	3+2+2+1+1+1
	3+3+2				3+2+1+1+1+1+1
8	3+2+2+1	9		10	3+1+1+1+1+1+1+1
	3+2+1+1+1				2+2+2+2+2
8	3+1+1+1+1+1	9		10	2+2+2+2+1+1
	2+2+2+2				2+2+2+1+1+1+1
8	2+2+2+1+1	9		10	2+2+1+1+1+1+1+1
	2+2+1+1+1+1				2+1+1+1+1+1+1+1
8	2+1+1+1+1+1+1	9		10	

Table 2.1 - CDA Module Coverage

Thus there are 94 possible combinations of multiple roots for polynomials of orders 4 to 10 that are covered by the CDA module.

Note that the method is applicable to not only polynomials with multiple + singular real roots, but also to polynomials with multiple + singular real and/or complex conjugate roots. Multiple complex conjugate roots however, will be found as singular pairs.

### 2.1.3 Time of Computation.

Where there are a large number of multiple roots, computation is relatively fast. Where there are a small number, or none, computation time is much longer. Times range from a few seconds to several minutes. In order to shorten computation time where possible, when a polynomial contains only integer coefficients, its roots will also be integers, and in this case the roots of the various differentials, are rounded to integers before trial division into the original polynomial. Should, by coincidence, such a rounded root correspond to a root, multiple or singular, of the original equation, computation time is thereby considerably reduced. Accordingly, this will cause single roots to be found during the multiple search, and thus displayed in the multiple root output message. In the implementation here, the roots of the derivatives are found using the Bairstow module.

Where it is known that a particular equation definitely does not contain multiple roots, the facility to disable the CDA module has been provided. In such a case the equation to be analysed is passed directly to the Bairstow module. The facility to disable the DDR module has also been added in this update, although this is not to shorten computation time, the time gained is minimal, but rather to enable comparison of the various methods in terms of accuracy. The modes of operation possible are summarised in the following table.

<b>DDR Module</b>	<b>CDA Module</b>	<b>Bairstow Module</b>
Active	Active	Active
Disabled	Active	Active
Active	Disabled	Active
Disabled	Disabled	Active

**Table 2.2 - Operational Modes.**

### 2.1.4 Accuracy.

Implementation of this module via an Excel spreadsheet, suffers from the same limitation as the Bairstow and DDR modules as delineated in [1] and [2]. That is, that Excel only computes to 15 significant places. Hence, when dealing with numbers greater than this, errors will result. This is particularly so when performing the division of trial roots into the polynomial under analysis. To alleviate this as much as possible, a set of algorithms has been created to augment the decision making process, as to whether any particular trial root, produces a zero remainder when divided into the original polynomial. While these work very satisfactorily when the multiple and other roots are well spaced, they are not always fully successful when the roots are very close together. Thus, when this appears to be the case, it is always wise to check that, when multiplied up, the roots so obtained produce the coefficients of the original equation. This is most easily done by entering the roots into the Polynomial Construction spreadsheet that has also been provided in BAIRSTOW3.ZIP. Where necessary, the roots can then be iterated via the facility provided on that spreadsheet, until coincidence with the coefficients of the original polynomial is achieved. This is not always an easy task and so, to this end, the Appendix to this paper provides some brief guidelines on performing this process efficiently.

### 3 Conclusions

The preparation of this paper, and the addition of the CDA module to the DDR and Bairstow modules in BAIRSTOW3.XLS, completes this combination of polynomial root finding methods. This final spreadsheet, together with the EQHOLD.XLS and POLYNOMIALCONSTRUCTION.XLS spreadsheets, provides the means by which the roots of any polynomial, with real, complex or both coefficients, can be precisely determined. In most cases the roots so found will be precise, (to four decimal places), but where the roots are very close together, the BAIRSTOW3.XLS spreadsheet alone can produce small errors. These can however, be eliminated via the root iteration process in the Polynomial Construction spreadsheet, so that in all cases, precise roots can be obtained.

It should be noted that in BAIRSTOW3.XLS the DDR module has been updated to remove the erroneous identification, and premature stop, when some of the coefficients of the polynomial under test are zero, or when one or more of the roots are zero.

#### APPENDIX A.

##### Root Iteration.

As previously stated, when the actual roots of a polynomial are very close together, and/or are very large, their extraction by any root finding technique can result in small errors. These errors can be removed by comparing the coefficients of the polynomial the trial roots produce, with those of the original equation, and iterating the trial roots until the two sets of coefficients coincide. The illustration of a technique to accomplish this is best effected by an example using the Polynomial Construction spreadsheet.

Consider the following 7th order equation.

$$\begin{aligned}
 &x^7 + (9.5870E + 03)x^6 + (3.9038E + 07)x^5 + (8.7651E + 10)x^4 \\
 &+ (1.1733E + 14)x^3 + (9.3728E + 16)x^2 + (4.1399E + 19)x + 7.8045E + 21
 \end{aligned}
 \tag{A.1}$$

This equation is known to contain only singular roots. To set up the example, insert the coefficients of (A.1) into the "Original Equation" cells on the Polynomial Construction spreadsheet.

This polynomial, when run through the Bairstow module, on the BAIRSTOW3.XLS spreadsheet, with the DDR and CDA modules disabled, results in the following set of trial roots.

$$\begin{aligned}
 r_1 &= 1231.9583 \\
 r_2 &= 1248.5489 \\
 r_3 &= 1243.9002 + j4.8481 \\
 r_4 &= 1243.9002 - j4.8481 \\
 r_5 &= 1235.8462 + j5.0832 \\
 r_6 &= 1235.8462 - j5.0832 \\
 r_7 &= 2147.0000
 \end{aligned}
 \tag{A.2}$$

Because the coefficients of the original equation are all integers, so then will its roots be all integers. Root  $r_7$  is therefore at this stage assumed to be correct. The remaining values in (A.2) can therefore be rounded to integers to give, in ascending order

$$\begin{aligned}
r_1 &= 1232 \\
r_2 &= 1235 + j5 \\
r_3 &= 1235 - j5 \\
r_4 &= 1244 + j5 \\
r_5 &= 1244 - j5 \\
r_6 &= 1249 \\
r_7 &= 2147
\end{aligned} \tag{A.3}$$

Now insert these rounded trial root values into the "Root Input" cells on the spreadsheet. The resulting trial polynomial coefficients are displayed in the row for order 7. The difference between the coefficients of the original equation and those of the trial equation,  $a_q(Diff)$ , are now displayed in the row below the original equation coefficients. The trial roots can now be iterated to reduce the difference row to zero. It is only necessary to reduce coefficients  $a_{(n-1)}(Diff)$  and  $a_{(n-2)}(Diff)$  to zero, all the rest will then also go to zero. It is important to note that, when  $a_q(Diff)$  is positive, the trial roots are too low and need to be increased and vice-versa when  $a_q(Diff)$  is negative.

In the example note initially that

$$\begin{aligned}
a_6(Diff) &= -1 \\
a_5(Diff) &= -8345
\end{aligned} \tag{A.4}$$

Insert the number 1 into the root iteration cell. The iteration process now starts as follows.

(i) First, to deal with the imaginary components of  $r_2, r_3, r_4$ , and  $r_5$ . These must be iterated so as to maximise  $a_5(Diff)$ . Increasing them reduces  $a_5(Diff)$  so that is in the wrong direction. Reducing them increases  $a_5(Diff)$  which maximises when these components are all zero. Thus there are no complex conjugate roots in the original equation. Note, this reduction iteration is effected by subtracting from  $jr_2$  and  $jr_4$  and adding to  $jr_3$  and  $jr_5$ , (ignore the EXCEL error "&value" in the "Difference" row while iterating these components.). This iteration results in

$$\begin{aligned}
a_6(Diff) &= -1 \\
a_5(Diff) &= -8295
\end{aligned} \tag{A.5}$$

(ii) The coefficient  $a_6$  is the sum of all the roots, so that to reduce  $a_6(Diff)$  to zero, it is here only necessary to reduce one of the trial roots by unity. Therefore reduce  $r_1$  by unity to give

$$\begin{aligned}
a_6(Diff) &= 0 \\
a_5(Diff) &= 61
\end{aligned} \tag{A.6}$$

(iii) Because it is known that the original polynomial contains only singular roots, it is now necessary to separate  $r_2$  &  $r_3$  and  $r_4$  &  $r_5$ . This must be done while maintaining  $a_6(Diff)$  at zero, and also reducing  $a_5(Diff)$ . Therefore increase  $r_3$  by unity and reduce  $r_4$  by unity. This gives

$$a_5(Diff) = 54 \tag{A.7}$$

(iv) Now begins the process of reducing  $a_5(Diff)$  to zero. The coefficient  $a_5$  is the sum of the product of the roots taken in pairs, so this process deals with the trial roots in pairs. Thus because  $a_5(Diff)$  is positive, increase a low root and decrease a high root until  $a_5(Diff)$  just goes negative, (avoid incurring duplicated roots). Consequently, in unity steps each, increase  $r_1$  by 4 and decrease  $r_6$  by 4. This gives

$$a_5(Diff) = -2 \tag{A.8}$$

(v) Now increase a high root and decrease a low root, (other than  $r_1$  in conjunction with  $r_6$ ), until  $a_5(Diff)$  just goes positive, (avoid incurring duplicated roots). Therefore decrease  $r_4$  by unity and increase  $r_6$  by unity. This gives

$$a_5(Diff) = +1 \quad (\text{A.9})$$

(vi) Repeat (iv), (other than  $r_1$  &  $r_6$  and  $r_4$  &  $r_6$ ). Therefore increase  $r_3$  by unity and decrease  $r_4$  by unity. This gives

$$a_5(Diff) = -3 \quad (\text{A.10})$$

(vii) Repeat (v), (other than  $r_1$  &  $r_6$ ,  $r_4$  &  $r_6$  and  $r_3$  &  $r_4$ ). Therefore reduce  $r_5$  by unity and increase  $r_6$  by unity. This gives

$$a_5(Diff) = 0 \quad (\text{A.11})$$

All other coefficient differences are now also zero and the iterated roots are the true roots of (A.1).

This is just one of a number of iteration paths that result in determining the correct roots. Other paths may require a larger or smaller number of iterations.

When the coefficients of a polynomial are non-integers, so then will its roots be, and an iteration process on them will be a little more involved, but the same general principles will still apply. In performing processes (iii) to (vii) etc, start with an iteration value of unity and work until  $a_{(n-2)}(Diff)$  becomes less than  $\pm 1$ . Change the iteration value to 0.1 and work until  $a_{(n-2)}(Diff)$  falls below  $\pm 0.1$ . Continue this process until  $a_{(n-2)}(Diff)$  falls to zero or is minimised with an iteration value of 0.0001.

There is a small possibility that the iteration process may result in both  $a_{(n-1)}(Diff)$  and  $a_{(n-2)}(Diff)$  going to zero when all others do not. In this case the iterated roots are not correct, and the iteration process should be backtracked to a suitable point to take a different path.

Finally, there is also a small possibility that, both  $a_{(n-1)}(Diff)$  and  $a_{(n-2)}(Diff)$  may go to zero, together with all other coefficient differences except one, which may still show a small residual. This is due to EXCEL's computational limitation as discussed in this paper and elsewhere, [1] and [2]. This residue may be eliminated by causing EXCEL to perform a re-calculation, i.e. by iterating a single root up and down etc.

## REFERENCES.

- [1] P.G.Bass, *Ensuring the Absolute Stability of the Bairstow Polynomial Root Extraction Method*, [www.relativitydomains.com](http://www.relativitydomains.com).
- [2] P.G.Bass, *Precise Determination of the Multiple Roots of High Order Polynomials - 1 - The Differential Division Remainder Method*, [www.relativitydomains.com](http://www.relativitydomains.com).