ENSURING THE ABSOLUTE STABILITY OF THE BAIRSTOW POLYNOMIAL ROOT EXTRACTION METHOD.

Peter G.Bass.

ABSTRACT.

Bairstow's Method of finding the roots of polynomial equations is examined in detail to determine the ways in which it fails to produce a satisfactory result. These problems are then eliminated in an experimental computer spreadsheet implementation, (Microsoft EXCEL), for polynomials of orders up to 10.

1 INTRODUCTION.

There are many methods of determining the roots of linear and non-linear equations of a single variable as delineated in [1]. Perhaps the most widely known is the Newton-Raphson Method, within a population that ranges from the very simple Bi-Section Method, up to the sophisticated Jenkins-Traub Method, which can deal directly with equations with complex coefficients.

All of these methods use a process of iteration starting from one or more "initial guesses", to home in on one or two roots simultaneously, and then reducing the original equation accordingly to repeat the process until all the roots have been found. They all enjoy varying degrees of success, depending upon their sophistication and complexity. However, they are also subject to a number of failure mechanisms, and restrictions in application, and there does not appear to be one single process that will reliably deal with all types of equations.

The method to be studied here is known as Bairstow's Method for polynomials, and was developed by Sir Leonard Bairstow and first published in an Appendix in his book, "Applied Aerodynamics" in 1920. This method is particularly appealing because its convergence is quadratic, and its mathematical technique can easily be implemented in a computer spreadsheet, so eliminating a considerable amount of computer code development. However, Bairstow's Method also suffers from a number of restrictions and failure mechanisms, which have resulted in it being of limited use.

It is the purpose of this paper to analyse these problems and eliminate them in two experimental spreadsheets that can be downloaded from the website as a ZIP file. Of the two spreadsheets provided, the first, "Bairstow.xls", is the main one which will determine the roots of any polynomial with real coefficients up to order 10, from the input of its coefficients. It is similar to, but more extensive than, the spreadsheet application of Bairstow's Method presented in [2]. The second spreadsheet, "Polynomial Construction.xls", serves three purposes, (i) construction of any polynomial up to order 10 from the input of its roots, both real and complex, (ii) iteration of the roots of a polynomial to make minor adjustments to its generated coefficients, and (iii) the multiplication together of two polynomials each of order up to 5, again with both real and complex coefficients. The in depth details and usage of both of these spreadsheets is described more fully in the later text.

2 Bairstow's Method of Finding the Roots of Polynomial Equations.

2.1 A Brief Description of Methodology.

In Bairstow's Method, the equation to be solved is divided by a quadratic, the coefficients, r and s, of which, are initially specified as the "input guesses". This results in a reduced polynomial and a remainder. Based upon the coefficients of the original polynomial, and those of the dividing quadratic, the parameters r and s are then adjusted to new values using a Taylor's series expansion, with the objective of reducing the remainder to zero. This process is continuously repeated until the division remainder approaches zero to within a specified limit. The resulting roots of the dividing quadratic are then taken as the same as two of those of the original polynomial. The latter is then reduced by the roots found, and the whole process repeated until all the roots have been determined.
A more detailed description may be found in [1].

2.2 Failure Mechanisms and Restrictions.

As it stands there are basically four problems with this method, as follows.

(i) If the "initial guesses" of the coefficients of the dividing quadratic are unsuitable, the method can diverge. The reason for this is analysed in Appendix A.

(ii) If the required output precision is too tight, the method can hunt and fail to converge to a solution. The reason for this is also analysed in Appendix A.

(iii) The method is only applicable to polynomials with real coefficients. It can however, deal with these to any exponent order.

(iv) When the polynomial to be factored contains multiple identical or very close roots, the method can produce results that are outside the required precision.

These problems limit the applicability of the method, especially when high order polynomials need to be factored. However, because the method is most conducive to computer implementation in a spreadsheet, these problems can be eliminated. The first three problems are easily eliminated. The fourth is more difficult.

2.3 Problem Elimination.

The solutions to the problems described here, are as incorporated in the computer spreadsheet implementations associated with this paper.

2.3.1 Divergence.

The problem of divergence is eliminated by monitoring the remainder after each fourth division of the iterated quadratic, to see whether it is increasing or decreasing. If it is increasing, and continues to do so to exceed a predetermined limit, the original value of s is adjusted by unity, and the whole process restarted. This is repeated until the remainder stops diverging, and converges towards zero. The limit set in the Bairstow spreadsheet is 1E ± 100. See Appendix A.

2.3.2 **Hunting.**

To eliminate the problem of hunting, first note that precision is initially specified by how close the remainder should approach zero before the root is taken. The value specified in the Bairstow spreadsheet is 1E - 10. To avoid hunting, if the roots have not been found before 100 iteration cycles, this limit is continuously reduced by a multiplicative factor of 1.01, until it reaches the level of precision to which the method has achieved. The roots are then taken, the limit reset to the above value, and the process restarted to find the next pair of roots. See Appendix A.

2.3.3 **Complex Coefficients.**

The fact that the Bairstow Method is only applicable to polynomials with real coefficients is a serious limitation. The method itself cannot be modified to accommodate polynomials with such characteristics, but these types of equations can themselves be reformatted to make them applicable.
Consider the following generalised equation,

$$y = \sum_{i=0}^{n} A_i x^i \tag{2.1}$$

where

$$A_i = a_i + jb_i \tag{2.2}$$

To find the roots of (2.1) using Bairstow's Method, first multiply (2.1) by its complex conjugate equation, i.e.

$$y_c = \sum_{k=0}^{n} B_k x^k \tag{2.3}$$

where

$$B_k = a_k - jb_k \tag{2.4}$$

This results in,

$$yy_c = \sum_{i=0,k=0}^{n} A_{(n-i)} B_{(n-k)} x^{(2n-i-k)} \tag{2.5}$$

and where all the coefficients are real. Equate (2.5) to zero and normalise the coefficient of the highest exponent to unity. Eq.(2.1) has thereby been converted to a polynomial of twice the order, but with real coefficients. The roots of (2.5) will be the complex conjugate pairs of (2.1) and (2.3), and any real roots in (2.1) and (2.3) will therefore be duplicated in the roots of (2.5).
To identify which roots of (2.5) belong to (2.1), first the number of all real roots will obviously be one half of all duplicated real roots. The complex roots can then be identified by applying Descartes' Rule of Signs to the imaginary coefficients of (2.1). This latter process may require a trial reconstruction of (2.1) from selected pairs of roots of (2.5), if the roots of (2.1) contain both positive and negative imaginary components.
Simple examples of the above procedure are given in Appendix B.
Note that the process of multiplying high order polynomials with complex conjugate coefficients, has been facilitated on the second sheet of the Polynomial Construction spreadsheet associated with this paper.

2.3.4 Multiple Roots.

When all the roots are real and identical, they can easily be determined from the appropriate comparison of $a_{(n-1)}$, the coefficient of $x^{(n-1)}$, and a_0, the constant term in the polynomial. Apart from this one case, this is the most difficult problem associated with any root extraction algorithm. Multiple roots not being correctly identified, and appearing as slightly separated singular roots. Sometimes small conjugate imaginary components also appear, when only real roots are expected. This can also occur when two roots are not quite identical, but very close. The worst case occurs when there (n - 1) identical roots, and one singular root, especially when the multiples and the singular are widely separated. To avoid this specific case, a new algorithm, independent of the Bairstow Method, has been developed that will identify all such roots with 100% accuracy and precision. The algorithm is based upon a specific relationship, that in this case, exists between the original polynomial and its first derivative.

In all remaining cases, extraction of all multiple root combinations are subject to the Bairstow Method as modified herein. Accordingly, if the number of identical roots is small, typically less than half the total number of roots, they are generally identified correctly. It is believed that this is because the method, being quadratically convergent, each half of an identical pair of roots is located in conjunction with one of the singular roots. If however, the number of identical roots exceeds one half of the total number of roots, this does not apply, especially when one of the multiple roots is the first to be taken. The problem appears to be due to the fact that with identical roots, when the polynomial is graphed, the curve does not cross the x axis, but just touches it at one singular point, plus the nature of iterative analysis being a non-continuous process, the method is unable to converge to the exact root. The difficulty is partly alleviated in the Bairstow spreadsheet application associated with this paper, by subjecting every reduced equation to a new part multiplicity root check algorithm before continuing with the Bairstow analysis. This produces a good result when all the singular roots have been identified, and only multiple roots are left in the reduced equation. Where this is not the case, the remainder of the division process oscillates close to zero at the location of the remaining multiple roots by an amount determined by the iteration amplitude. In the application here, this continues until the hunting avoidance mechanism reduces the required precision to that attained by the iteration process and the "roots" are then taken. Errors with multiple complex roots are not so great because of the variation resulting from the opposing imaginary components.

The Bairstow Method cannot be modified to eliminate this difficulty, and so, another method of elimination must be used. The method selected here, which applies to multiple complex roots as well as real, is as follows.
First, a simple algorithm has been included in the Bairstow spreadsheet to indicate when multiple root errors may be present in the results, and the multiple roots thereby detected as very close singular roots, straddling them with errors of up to ±3%. Thus when this is the case, all the roots found can be copied to the Polynomial Construction spreadsheet, and inserted into the input cells in ascending order. The resulting polynomial coefficients can then be directly compared with those of the original equation, for which appropriate input cells have been provided. If there are discrepancies, the individual roots can then be iterated by any amount, up or down, until the resulting polynomial coefficients exactly match those of the original equation. The final roots are then precise.

When the multiple roots involved are integer, or posses just a small number of decimal places, this process is relatively easy and a satisfactory result achieved quite quickly. If the multiple roots are complex, and/or involving very high precision, the iteration process will be more difficult and require some patience and good interpretative skills.

2.4 Computer Spreadsheet Implementations.

2.4.1 The Bairstow Spreadsheet.

The Bairstow spreadsheet comprises four separate sheets. The first contains basic instructions, an area for the input of polynomial coefficients, and an area for the display of the roots found, with real and imaginary components presented in the form $a_i \quad \pm j\ b_i$.
The input of initial guesses of r and s are not necessary because, via testing, optimum values have been found for a very wide range of polynomial characteristics, and are selected according to the coefficient inputs. However, a facility for the manual input of r and s has been provided should it be so required.
Also included in this sheet, is the possible multiple root error warning message, shown just below the root output display.
The second sheet in this spreadsheet is the Bairstow process calculation sheet which does not require user intervention. It is displayed during operation, where it shows not only the root determination process in operation, but also the part multiplicity root check algorithm, and the divergence, hunting and reduced equation corruption avoidance mechanisms, (for this last, see Section 2.4.4). Note that sometimes during operation, this sheet appears to "freeze", with only the cycle counter and the accuracy limit counter operating. This is normal and means that the iteration amplitude has reduced to zero, and the accuracy limit is being counted down to meet that attained by the iteration process, i.e. the hunting avoidance mechanism.
The third sheet contains the new algorithm for the extraction of the (n- 1) multiples plus one singular root.
Finally, the fourth sheet contains further information and instructions, the latter primarily the procedure to be adopted when possible multiple root errors have been indicated in the results. A disclaimer on the use of this spreadsheet is also included here.

2.4.2 The Polynomial Construction Spreadsheet.

This spreadsheet comprises three separate sheets. The first will generate any polynomial up to order 10 from the input of its real/complex roots. This sheet is provided primarily for twofold use. First, in removing multiple root errors in those roots found via the Bairstow spreadsheet. To that end a facility has been included to enable the detailed and precise iteration, positive or negative, of any root component. This process allows the reformulation of the coefficients of the original equation, so removing multiple root errors and thereby obtaining a set of very precise roots.
The second purpose of this sheet is to determine, again from a reconstruction of the original polynomial, the correct roots of an equation with complex coefficients when it contains roots with both positive and negative imaginary components
The second sheet deals with the multiplication of any two polynomials of order 1 to 5, with either real or complex coefficients. It is primarily provided to enable multiplication of complex conjugate polynomials, so that equations with complex coefficients can be analysed in the Bairstow spreadsheet for their roots.
The third sheet once again contains further information, instructions and a disclaimer.
Both of the above spreadsheets can be downloaded here as a ZIP file, Bairstow.zip. Microsoft EXCEL 97 or later is needed to run them.

2.4.3 Testing.

The testing discussed here concerns only the Bairstow spreadsheet. It is emphasised that it is of course not possible to test all configurations of polynomials, there being for even just quadratics,

an infinite number etc. The results quoted below can therefore only be considered as typical, and variations may be experienced either up or down depending upon any polynomial analysed.
When just singular roots are involved, they were all correctly identified to an accuracy of at least 0.001%, albeit only to the displayed four decimal places. The cases tested were a combination of very small, very large and a mixture of same, covering both real and complex conjugate roots.
The major part of the test programme was concerned with polynomials that contained a multiplicity of identical roots. In the group of polynomials of orders 3 to 10 inclusive, there are 127 possible combinations of multiple roots. All of these were tested with both integer and four decimal place roots with results as shown in the following table.

Order	**No. of Combinations of Multiple Roots**	**Max. % Error (Magnitude)**	**Multiple Root Combination with Max. Error**
3	2	0	No Errors
4	4	0	No Errors
5	6	0.010	3+1+1
6	10	0.094	4+1+1
7	14	0.085	5+1+1
8	21	2.17	6+2
9	29	2.04	5+4
10	40	3.29	8+1+1
Total	127	-	-

Table 2.1 - Test Results for Multiple Roots.

Note: The nomenclature 3+1+1 means a triple root plus two singles etc.
All of the above discrepancies were able to be eliminated via the Polynomial Construction facility. Note that where there were n multiple roots, no errors were encountered, and with respect to the new algorithm for (n - 1) multiple roots plus a single, again no errors resulted.

2.4.4 Limitations.

Apart from the obvious limitations deliberately included of displayed output to a maximum of 4 decimal places, and the maximum order of polynomial being 10, (5 for those with complex coefficients), there are two other limitations of the application presented here.
When the roots are greatly separated, the method can give fictitious results. This is because if a very large root is taken in conjunction with a very small one, the accuracy with which the small root is taken is governed by that applicable to the large. This can cause the small root to be very inaccurate, which in turn can corrupt the reduced equation so resulting in the rest of the roots also being grossly wrong. To avoid this wherever possible, a method of detecting corruption of the reduced equation has been incorporated. When this is triggered, the process is halted, the values of r and s varied, and the process re-initialised and re-started. In extreme cases this form of corruption can persist after adjustment of r and s. In such cases the avoidance procedure will trigger for a maximum of 10 times with increasingly large values of r and s adjustment. Subsequently, the root finding process is allowed to continue. In these cases the results should be carefully checked via the Polynomial Construction spreadsheet facility to ensure that the roots found are good. With

regard to this feature, it is also very important to note that, there are a number of polynomials which inherently exhibit the criteria used here to test for corruption of a reduced equation. i.e. equations containing complex roots or real roots with mixed parity. These equations will, after the above 10 times test, be factored correctly. The penalty incurred by way of delay in these equations being analysed, is merely a matter of a few seconds.
The second limitation concerns the computational hardware. All computers are subject to very small rounding errors, and when computing with very large numbers, these can become significant. This limits the size of a_0 in this application to between 1E-9 and 1E+40. Outside of this range, accuracy may in some cases be in excess of that quoted in Table 2.1. A warning message is shown accordingly.

3 Conclusions.

The enhancements of the basic Bairstow Method of polynomial root extraction introduced here has, it is believed, improved both its functionality and its applicability. In its spreadsheet application, elimination of the divergence and hunting problems have ensured that it will always converge to a solution. Accuracy has in some cases been improved by the introduction of the algorithms to identify both n, and (n - 1) identical roots, and the part multiplicity root algorithm. However, this still leaves the inaccuracies for all other configurations of multiple roots, albeit they can be corrected via the Polynomial Construction spreadsheet facility. This however, in the case of complex roots, and/or high precision, can be a difficult and time consuming task, and ideally, if these roots could be detected accurately, could be dispensed with.
Further improvement can be achieved in essentially three ways, as follows,

(i) Improving accuracy for all other configurations of multiple roots. Clearly this is the most important improvement target, and further algorithms for this purpose are being developed.

(ii) Extending the order of polynomials that can be handled. This is merely an extension of spreadsheet and macro coding. However, as the order is increased, the amount of extra coding in the Polynomial Construction spreadsheet becomes, for one individual, prohibitively high.

(iii) Extending the precision of the roots found. This can initially be achieved by simply increasing the number of decimal points displayed in the output. However, if very high precision were required, it would need modification of Bairstow's original concept to include the second order terms in the Taylor series expansion of r and s. This has been suggested elsewhere.

Appendix A.

The Cause of Divergence and Hunting.

A.1 Divergence.

In the development of the Bairstow Method, the iterated values of the remainder coefficients are given by a Taylor series expansion, thus

$$\begin{aligned} b_1(r+\Delta r, s+\Delta s) &= b_1(r,s) + \Delta r\frac{\partial b_1(r,s)}{\partial r} + \Delta s\frac{\partial b_1(r,s)}{\partial s} + \cdots \\ b_0(r+\Delta r, s+\Delta s) &= b_0(r,s) + \Delta r\frac{\partial b_0(r,s)}{\partial r} + \Delta s\frac{\partial b_0(r,s)}{\partial s} + \cdots \end{aligned} \tag{A.1}$$

where $b_1(\ )$ and $b_0(\ )$ are the coefficient of x and the constant term in the division remainder. This leads to

$$\begin{aligned} b_1(r+\Delta r, s+\Delta s) &= b_1(r,s) + c_2\Delta r + c_3\Delta s + \varepsilon_1 \\ b_0(r+\Delta r, s+\Delta s) &= b_0(r,s) + c_1\Delta r + c_2\Delta s + \varepsilon_2 \end{aligned} \tag{A.2}$$

where c_1, c_2 and c_3 are functions of r and s and the coefficients of the original equation, and ε_1 and $\varepsilon_y$gre second and higher order terms in the Taylor series expansion. If the method is to converge, the term on the LHS must tend to zero. The values of Δr and Δs are obtained by ignoring the ε terms and assuming the LHS of (A.2) is indeed zero. This then leads to,

$$\begin{aligned} \Delta s &= \frac{c_1 b_1(r,s) - c_2 b_0(r,s)}{c_2^2 - c_1 c_3} \\ \Delta r &= \frac{c_2 b_0(r,s) - c_3 b_1(r,s)}{c_2^2 - c_1 c_3} \end{aligned} \tag{A.3}$$

Substituting these terms back into (A.2) then gives

$$\begin{aligned} b_1(r+\Delta r, s+\Delta s) &= \varepsilon_1 \\ b_0(r+\Delta r, s+\Delta s) &= \varepsilon_2 \end{aligned} \tag{A.4}$$

So that, possible divergence is apparently due to the second and higher order terms in the Taylor series expansion becoming dominant. These terms are higher order functions of r and s, and the coefficients of the original equation, which are fixed.
Let the ε terms here be given by just the second order terms in the Taylor series expansion, then

$$\begin{aligned} \varepsilon_1 &= \frac{(\Delta r)^2}{2!}\frac{\partial^2 b_1(r,s)}{\partial r^2} + \frac{(\Delta s)^2}{2!}\frac{\partial^2 b_1(r,s)}{\partial s^2} \\ \varepsilon_2 &= \frac{(\Delta r)^2}{2!}\frac{\partial^2 b_0(r,s)}{\partial r^2} + \frac{(\Delta s)^2}{2!}\frac{\partial^2 b_0(r,s)}{\partial s^2} \end{aligned} \tag{A.5}$$

For a quartic polynomial, in which the coefficient of the highest exponent has been normalised to unity

$$\begin{aligned} \frac{\partial b_1(r,s)}{\partial s} &= a_3 + 2r = c_3 \\ \frac{\partial b_1(r,s)}{\partial r} &= \tfrac{\partial b_0(r,s)}{\partial s} = a_2 + 2s + 2ra_3 + 3r^2 = c_2 \\ \frac{\partial b_0(r,s)}{\partial r} &= a_1 + 2sa_3 + 6rs + 2ra_2 + 3r^2a_3 + 4r^3 = c_1 \end{aligned} \tag{A.6}$$

So that

$$\begin{aligned}
\frac{\partial^2 b_1(r,s)}{\partial s^2} &= \frac{\partial c_3}{\partial s} = 0 \\
\frac{\partial^2 b_1(r,s)}{\partial r^2} &= \frac{\partial c_2}{\partial r} = 2a_3 + 6r \\
\frac{\partial^2 b_0(r,s)}{\partial s^2} &= \frac{\partial c_2}{\partial s} = 2 \\
\frac{\partial^2 b_0(r,s)}{\partial r^2} &= \frac{\partial c_1}{\partial r} = 6s + 2a_2 + 6ra_3 + 12r^2
\end{aligned} \tag{A.7}$$

where in (A.6) and (A.7) the "a" parameters are the coefficients of the original equation.
Substitution of (A.7) into (A.5) then gives

$$\begin{aligned}
\varepsilon_1 &= \frac{(\Delta r)^2}{2!}(2a_3 + 6r) \\
\varepsilon_2 &= \frac{(\Delta r)^2}{2!}\left(6s + 2a_2 + 6ra_3 + 12r^2\right) + (\Delta s)^2
\end{aligned} \tag{A.8}$$

In (A.8), the coefficients of $(\Delta r)^2$ and $(\Delta s)^2$ are finite and from their form cannot be the source of divergence. Thus if divergence occurs it must be because Δr and Δs themselves diverge.
From (A.3) these terms will only diverge if

$$c_2^2 - c_1 c_3 \to 0 \tag{A.9}$$

Substitution of (A.6) into (A.9) yields

$$\begin{aligned}
c_2^2 - c_1 c_3 = {} & r^4 + 8a_3 r^3 + \left(6s + 5a_2 + 5a_3^2\right) r^2 \\
& + \left(4a_3 s + 2a_2 a_3 - 2a_3^2 s - 6a_2 s - a_1 a_3\right) r \\
& + 4a_2 s + a_2^2 - 2a_3^2 s - a_1 a_3
\end{aligned} \tag{A.10}$$

and thus divergence can occur via the iteration process when r approaches any of the roots of (A.10). These roots clearly depend upon the value of s, and in the divergence avoider in this application therefore, it is sufficient, when necessary, to vary only s by a small amount to move r away from these roots, and ensure that the method converges.
Also note that with Δr and Δs as the ultimate source of divergence, the primary terms in (A.2) are also subject to this phenomenon and consequently, under these conditions, (A.4) is not a valid relationship.
For other higher order polynomials, the equivalent equation to (A.10) will be of the same order as the polynomial, giving more possibilities for divergence, but the same criteria applies with regard to avoidance.

A.2 Hunting.

In the absence of divergence, (A.4) is valid and, from this, ε_1 and ε_2 are clearly the minimum values to which $b_1(\ )$ and $b_0(\ )$ can reduce. The higher the order of the equation to be analysed, the larger these terms will be, and therefore the greater the potential error in the results, especially with multiple roots. Consequently, if these error terms are above the required precision, then the

method can never converge to a solution. The only recourse is therefore, to reduce the required precision until it reaches the higher of the two error terms above. This is the purpose of the hunting avoider.

Appendix B.

Solution of Polynomials with Complex Coefficients - Examples.

B.1 Same Sign Roots, (Imaginary Component).

Consider the equation,

$$y = (x+2)\{x+(1+j2)\}\{x+(3+j4)\} \tag{B.1}$$

This multiplies out to,

$$y = x^3 + (6+j6)x^2 + (3+j2)x + (-10+j20) \tag{B.2}$$

To determine the roots of (B.2) using Bairstow, first multiply (B.2) by its complex conjugate equation

$$y_c = x^3 + (6-j6)x^2 + (3-j2)x + (-10-j20) \tag{B.3}$$

To give

$$yy_c = x^6 + 12x^5 + 78x^4 + 280x^3 + 621x^2 + 820x + 500 \tag{B.4}$$

Equating (B.4) to zero and inserting its coefficients into the Bairstow spreadsheet yields the roots as,

$$yy_c = (x+2)^2\{x+(1+j2)\}\{x+(1-j2)\}\{x+(3+j4)\}\{x+(3-j4)\} \tag{B.5}$$

Consequently, within (B.5), the roots of (B.2), from an application Descartes' Rule of Signs to the imaginary components of (B.2), are determined to be those of (B.1).

B.2 Opposite Sign Roots, (Imaginary Components).

If (B.1) were

$$y = (x+2)\{x+(1+j2)\}\{x+(3-j4)\} \tag{B.6}$$

To give

$$y = x^3 + (6-j2)x^2 + (19-j2)x + (22+j4) \tag{B.7}$$

Multiplying (B.7) by its complex conjugate again gives (B.4) and subsequently (B.5). Descartes' Rule of Signs applied to (B.7) then shows complex roots with both positive and negative imaginary components. The correct ones are then identified by inserting the two complex root combination possibilities, i.e. (1+$j2$) with (3-$j4$) or (1-$j2$) with (3+$j4$), together with the single real root, into the Polynomial Construction spreadsheet to determine which gives the correct coefficients of (B.7).

References.

[1] *Root Finding Algorithms*, www.wikipedia.com.

[2] Karim Y Kalaban, Ali El-Hajj, Shahwan Khoury and Fadi Yousuf, *Root Computations of Real-Coefficient Polynomials Using Spreadsheets,* Inst. J. Engng Ed. Vol. 18 No. 1 pp 89-97, 2002.