Finding Roots by Deflated Polynomial Approximation

by T. N. LUCAS

Department of Mathematical and Computer Sciences, Dundee Institute of Technology, Bell Street, Dundee DD1 1HG, U.K.

ABSTRACT: *A numerical technique is presented which evaluates the roots of polynomials with real coefficients. Features of the method include no complex arithmetic requirements, no need to guess at initial quadratic factor estimates, multiple or nearly equal roots being easily dealt with and a high degree of flexibility in coping with non-convergent iterations. The method is simple to use and is based upon a Routh Array-type algorithm familiar to control engineers. Numerical examples demonstrate its application to various polynomials.*

I. Introduction

Finding roots of polynomials with real coefficients is a topic of great importance in many branches of science and engineering. For instance, the explicit solutions of linear, constant differential system models eventually come down to solving real polynomial equations. As such, considerable effort has been made in the past to develop efficient root-finding methods.

The techniques which have proved to be popular for polynomials in a general sense include (**1**) the Newton–Raphson method, Graeffe's "root-squaring" method, Lin's method, Bairstow's method and, to a lesser extent, the Secant and Bisection methods. Of course, all numerical methods have their drawbacks and the successful application of any of these methods depends on the particular distribution of the roots to be found. Consequently, it is not unusual to have to experiment with initial estimates or function evaluations before deciding upon a method.

One of the problems with real polynomials is that there may be complex conjugate root pairs, thus rendering the Secant and Bisection methods ineffective in general, and making the Newton–Raphson and root-squaring methods rather cumbersome to use. Both Bairstow's and Lin's methods are based upon iteratively finding quadratic factors of the polynomial and hence circumvent the problem of having to use complex arithmetic. As a result, these methods have proved very popular for solving polynomial equations, especially from an engineering viewpoint, with possibly Bairstow's method being generally preferred for its quicker convergence rate over Lin's method when good initial estimates are available for a quadratic factor. However, Lin's method is simpler to use than Bairstow's and is not so sensitive to the initial quadratic factor estimate (**1**). With the advent of digital computers, it has enjoyed widespread popularity in control engineering in

© The Franklin Institute 0016–0032/90 $3.00 + 0.00

particular and is widely used in pole/zero determination of system transfer functions (**2**), and related work. It is also interesting to note that Aitken (**3**) expresses a guarded preference for Lin's method over Bairstow's because of its relative simplicity.

In this paper we present a new iterative method for finding quadratic factors of real polynomials. It is as easy to use as Lin's method and appears to be more flexible in coping with non-convergent iterations. A big advantage of the technique over other methods is its ability to cope with multiple or nearly-equal roots, a direct result of using the derivative of the polynomial to start the iterative process which also eliminates the need to "guess" initial approximations to quadratic factors. The method, like those of Lin and Bairstow, makes use of a Routh Array-type iteration algorithm (**4, 5**) which is known to be essentially the "synthetic-division" or "Euclid" algorithm and is very amenable to computer implementation, or even hand calculator use.

An interesting link with the Newton–Raphson method is pointed out and numerical examples are included to illustrate the application of the method on various polynomials.

II. The Extended Remainder Theorem and Taylor Approximants

To develop the underlying ideas of the new method it is necessary first of all to clarify the well-known "remainder" result of division of polynomials in terms of multipoint Taylor polynomial approximations.

If $P(x)$ is an nth degree polynomial and $D(x)$ is an mth degree polynomial with $m < n$, then division from highest powers gives

$$\frac{P(x)}{D(x)} \equiv Q(x) + \frac{R(x)}{D(x)} \tag{1}$$

where $Q(x)$ is the $(n-m)$th degree quotient polynomial and $R(x)$ is the "proper" $(m-1)$th degree remainder polynomial.

Equation (1) can also be written as

$$P(x) \equiv Q(x)D(x) + R(x) \tag{2}$$

and suppose that $D(x)$ has roots at $x = x_i$ $(i = 1, 2, \ldots, m)$, then Eq. (2) gives

$$P(x_i) = R(x_i) \tag{3}$$

for $i = 1, 2, \ldots, m$ which, in turn, shows that $R(x)$ is the $(m-1)$th degree multipoint Taylor polynomial approximant of $P(x)$ about the points $x = x_i$ for $i = 1, 2, \ldots, m$.

The above interpretation gives an "extended" remainder theorem which can be stated as: the remainder polynomial obtained when dividing two polynomials is the multipoint Taylor approximant of the numerator polynomial about the points which are the roots of the denominator polynomial, i.e. the numerator and remainder are equal in value at the roots of the denominator. It should be noted that if $D(x)$ has a multiple root, of multiplicity r say $(r \leqslant m)$, then repeated differentiation

up to $(r-1)$ times of Eq. (2) and substitution of $x = x_1 = x_2 = \cdots = x_r$ will generalize Eq. (3) to be

$$\frac{d^j P}{dx^j}(x_i) = \frac{d^j R}{dx^j}(x_i) \quad i = 1, 2, \ldots, r, \quad j = 0, 1, \ldots, r-1$$

and

$$P(x_i) = R(x_i) \quad i = r+1, r+2, \ldots, m. \tag{4}$$

A similar result is seen to apply in cases with two or more multiple roots. It should also be noted that Eqs (3) and (4) are equally applicable to complex points.

As an example of demonstrating the property given in Eq. (4) consider dividing $P(x) = 3x^4 - 2x^3 + 4x^2 + 5x - 2$ by the cubic $D(x) = (x-1)^2(x-2) = x^3 - 4x^2 + 5x - 2$. This gives the result

$$\frac{P(x)}{D(x)} = 3x + 10 + \frac{29x^2 - 39x + 18}{x^3 - 4x^2 + 5x - 2}.$$

It is easily verified that $R(x) = 29x^2 - 39x + 18$ is equal to $P(x)$ at $x = 1$ and $x = 2$, also

$$\frac{dR}{dx}(x) = \frac{dP}{dx}(x) \quad \text{at} \quad x = 1.$$

III. Finding Roots Using Remainders

The interpretation of the extended remainder theorem given in the previous section now forms the basis of the new technique for finding roots of real polynomials. A similar approach to the methods of Lin and Bairstow will be followed in that a quadratic factor of the polynomial is iteratively found before repeating the procedure on the deflated polynomial until all such factors are found. However, unlike these methods, the new technique finds a quadratic factor by dividing the polynomial of degree n by one of degree $(n-1)$, such that the remainder is of degrec $(n-2)$ and is then taken as an estimate of the deflated polynomial. An advantage of using this idea is that only two division steps per iteration are needed, whereas dividing a trial quadratic factor requires more if $n > 3$.

Following the notation of Eq. (1), it is clear that $R(x)$ is now the Taylor approximant of $P(x)$ about the $(n-1)$ points which are the roots of $D(x)$. Consequently, $R(x)$ may be taken as an estimate of the deflated polynomial containing $(n-2)$ roots of $P(x)$. To improve upon this estimate, $P(x)$ is divided by the $(n-1)$th degree polynomial $(x-p)R(x)$, where p is some arbitrary number which can often be taken to be zero. Hence, the second estimate of the deflated $(n-2)$th degree polynomial will be the Taylor approximant of $P(x)$ about the roots of $(x-p)R(x)$. Repetition of this procedure gives the iterative formula

$$\frac{P}{(x-p)R^{(i)}} = Q^{(i+1)} + \frac{R^{(i+1)}}{(x-p)R^{(i)}} \tag{5}$$

$i = 1, 2, \ldots,$ with $R^{(1)} \equiv R(x)$ and the function notation has been omitted for convenience.

For a convergent iteration it is seen that

$$R^{(i)} \to R^{(i+1)} \to R_L \quad \text{and} \quad Q^{(i+1)} \to Q_L$$

for large enough i, giving, from Eq. (5)

$$P \equiv Q_L(x-p)R_L + R_L$$

and hence

$$\frac{P}{R_L} \equiv (x-p)Q_L + 1. \tag{6}$$

Notice that $P/R_L \equiv ax^2+bx+c$ is the first quadratic factor of $P(x)$ and so Eq. (6) gives

$$ax^2+bx+c \equiv (x-p)(ex+d)+1$$

where $Q_L \equiv ex+d$ is the limiting quotient term, which in turn gives

$$\left.\begin{aligned} a &= e \\ b &= d-pe \\ c &= 1-pd \end{aligned}\right\} \tag{7}$$

thus completely determining the first quadratic factor. If $n > 4$ then the above procedure is applied to the deflated polynomial $R_L(x)$ to find the next quadratic factor, and so on until all such factors have been extracted.

At this point some observations about the method need to be clarified before discussing the choice of divisor polynomials in the next section. The first thing to mention is that, like all iterative methods, convergence is not always guaranteed and depends upon the choice of the initial divisor $D(x)$ and the subsequent divisors $(x-p)R^{(i)}(x)$. However, because the parameter p can be any number then this gives the method a useful degree of control on the convergence of the iteration. It is noticed that the polynomial and its deflated estimates are matched at $x = p$; changing the value of p effectively changes the shape of the deflated polynomial curve, which is necessary to produce a convergent iteration from a previously divergent one. This advantage is demonstrated by a numerical example later. Secondly, the value of p must remain constant throughout a quadratic factor iteration for Eqs (6) and (7) to apply. A consequence of this is that convergence tends to follow a linear rather than quadratic pattern. Thirdly, all calculations may be conveniently carried out via a Routh Array-type algorithm, which is a tabular way of performing the algebraic division process (**4, 5**). This algorithm is well known to control engineers and is the main reason why it was adopted for this method. A description of the algorithm is given in the Appendix.

IV. Choice of Divisor Polynomials

In order to start the iterative process to extract a quadratic factor it is necessary to divide the nth degree polynomial $P(x)$ by one of degree $(n-1)$, i.e. $D(x)$. It is now proposed to use the derivative $dP(x)/dx$ as this divisor for the following reasons:

(i) the roots of $dP(x)/dx$ always lie within the convex hull of the roots of $P(x)$ (**6**), and so in many cases the deflated polynomial estimates given by the remainders will approximate well to $(n-2)$ of the roots of $P(x)$;
(ii) if $P(x)$ has a multiple (or nearly equal) root then $dP(x)/dx$ will also contain this root, thus the remainder term, which is the Taylor approximant of $P(x)$ about the roots of $dP(x)/dx$, immediately yields the same root and hence avoids the convergence problems of other methods in such cases;
(iii) $dP(x)/dx$ is easily calculated from $P(x)$.

To give a simple illustration of the iterative process, Fig. 1 shows the first few iterations approximating to the deflated polynomial. The polynomial is a cubic with real roots with $p = 0$ and initial divisor $dP(x)/dx$. In this case the deflated polynomial is linear and the quadratic factor will yield the two "outside" roots. However, to illustrate what effect varying p has on the iteration, Fig. 2 shows the same cubic but with p chosen to lie to the left of the first root. It is clear that this value of p will lead to the quadratic factor giving the two right-most roots.

It should be stated that, in common with all numerical methods, there are occasions when singularities might occur in the division algorithm. The main cause of this situation is that the remainder polynomial is not of degree one less than the divisor polynomial. However, the iteration may be continued by multiplying the remainder polynomial by an appropriate number of linear factors to bring it up to the required degree. Notice that these factors can be quite arbitrary but ensure that the polynomial and its next deflated estimate will be matched about the roots of the factors. A numerical example is given later illustrating this case.

Before demonstrating the method by numerical examples, it is interesting to point out a link with the well-known Newton–Raphson method. If, instead of using a divisor polynomial of degree $(n-1)$ one of degree 2 is used, then the remainder will be linear and its root will be an estimate to a root of the (deflated) polynomial. A root estimate given by the Newton–Raphson method can be thought of graphi-

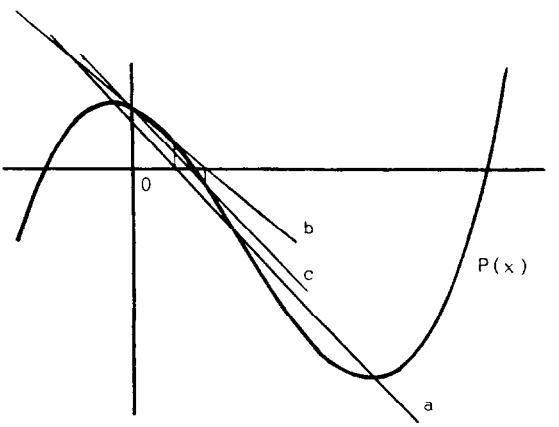

FIG. 1. a—1st deflated estimate, b—2nd deflated estimate, c—3rd deflated estimate.

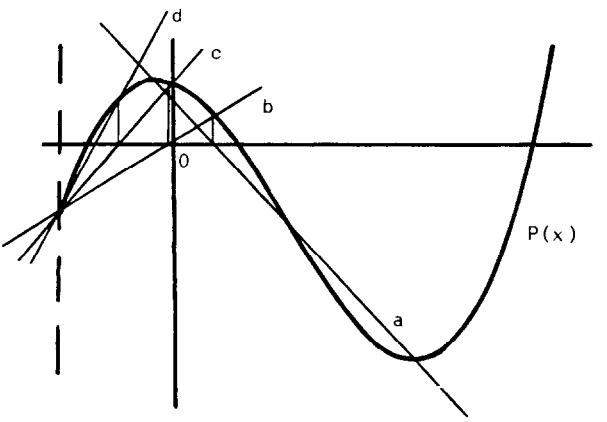

FIG. 2. a—1st deflated estimate, b—2nd deflated estimate, c—3rd deflated estimate, d—4th deflated estimate.

cally as the root of the linear Taylor approximant about a point (which is the previous root estimate), retaining the function value and its first derivative at this point. It was shown in Section II that this is equivalent to using a divisor of the type $(x-x_i)^2$ and hence the iteration becomes

$$\frac{P}{(x-x_i)^2} = Q^{(i+1)} + \frac{R^{(i+1)}}{(x-x_i)^2} \tag{8}$$

for some initial guess x_0 and $i = 0, 1, 2, \ldots$, where the root estimate x_i is the solution of the linear equation $R^{(i)}(x) = 0$.

For example, suppose it is known that a real root of $P(x) \equiv x^3 - 0.2x^2 - 0.2x - 1.2$ lies near 1.5. Choosing $x_0 = 1.5$ and using Eq. (8) gives the following table of values:

i	$R^{(i)}$	x_i
0		1.5
1	$5.95x-7.5$	1.26
2	$4.0588x-4.8832$	1.20
3	$3.64x-4.368$	1.20

It is easily verified that these iterates are equivalent to those of the Newton–Raphson method.

V. Numerical Examples

Example 1

To demonstrate the ability of the method to cope with the troublesome case of multiple/nearly equal roots, consider the polynomial

$$P(x) \equiv 16x^4 + 31.68x^3 - 8.8x^2 - 24.24x + 9.36.$$

Using the initial divisor $dP(x)/dx$ gives a quotient

$$Q^{(1)} = 0.25x + 0.12375$$

and remainder

$$\begin{aligned} R^{(1)} &= -16.1612x^2 - 16.002x + 12.3597 \\ &= -16.1612(x^2 + 0.9901x - 0.7648). \end{aligned}$$

Now choosing $p = 0$, the iteration given by Eq. (5) yields the quotients and remainders given in Table I.

It is seen that the sequence of quotients and remainders are converging and their corresponding coefficients agree to 2 decimal places after a few iterations. Note that it is not necessary to list both quotients and remainders for observing convergence because convergence of one implies convergence of the other. The quotient has only two terms and is thus simpler to use in this respect. Taking the last quotient estimate $Q^{(6)}(x)$ and using Eq. (7) gives a quadratic factor of $(-1.3046x^2 - 1.29x + 1)$ with roots at -1.500 and 0.511 to 3 d.p. The other quadratic factor is given by the remainder term $R^{(6)}(x)$, which has roots at 0.509 and -1.500. The original quartic has exact roots at -1.5, -1.5, 0.5 and 0.52 respectively. It is seen that the double root at -1.5 is given immediately by the method due to using the derivative polynomial as the initial divisor, and good approximations are obtained for the nearly equal roots; further iterations will give the required accuracy for these roots.

Example 2

Consider a typical control system (see Ref. (7)) with the 7th degree characteristic polynomial given by

$$P(x) \equiv x^7 + 83.64x^6 + 4097x^5 + 70342x^4 + 853703x^3 + 2814271x^2 + 3310875x + 281250.$$

The poles of the system, or the roots of $P(x)$, can be determined by applying the given method with $dP(x)/dx$ as the initial divisor and choosing $p = 0$. This gives

TABLE I

i	$Q^{(i)}(x)$	$R^{(i)}(x)$
2	$-0.99x - 0.98$	$-12.2452(x^2 + 0.9904x - 0.7644)$
3	$-1.3066x - 1.2930$	$-12.2515(x^2 + 0.9907x - 0.7640)$
4	$-1.3060x - 1.2920$	$-12.2578(x^2 + 0.9909x - 0.7636)$
5	$-1.3053x - 1.2910$	$-12.2640(x^2 + 0.9912x - 0.7632)$
6	$-1.3046x - 1.2900$	$-12.2702(x^2 + 0.9915x - 0.7628)$

the quotients listed in Table II. Using Eq. (7) with the coefficients given by $Q^{(8)}(x)$ in Table II gives the first quadratic factor (normalized) as $(x^2+64.1521x+2537.6256)$, with complex roots at $-32.076\pm38.843i$.

The corresponding remainder term,

$$R^{(8)}(x) \equiv x^5+19.4895x^4+308.6509x^3+1075.8711x^2+1301.6745x+110.8117,$$

is now the deflated polynomial containing the rest of the roots of $P(x)$. Again, using the iteration Eq. (5) with $P(x)$ replaced by its deflated estimate $R^{(8)}(x)$, initial divisor $dR^{(8)}(x)/dx$ and $p=0$ gives the quotients listed in Table III. Again using Eq. (7) with $Q^{(6)}(x)$ in Table III gives the next quadratic factor $(x^2+15.3473x+239.6919)$ with complex roots at $-7.674\pm13.447i$.

The remainder term given by

$$R^{(6)} \equiv x^3+4.1407x^2+5.4009x+0.4623$$

is the deflated cubic which is treated in the same manner as above giving, with

TABLE II

	$Q^{(i)}(x) \equiv ex+d$	
i	e	d
1	0.1429	1.7069
2	0.0032	0.3153
3	0.0002	0.0125
4	0.0003	0.0232
5	0.0004	0.0241
6	0.0004	0.0249
7	0.0004	0.0252
8	0.0004	0.0253
9	0.0004	0.0253

TABLE III

	$Q^{(i)} \equiv ex+d$	
i	e	d
1	1.0000	3.8976
2	0.0160	0.3303
3	0.0029	0.0457
4	0.0041	0.0629
5	0.0042	0.0639
6	0.0042	0.0640
7	0.0042	0.0640

$p = 0$, the quotients in Table IV. Equation (7) and $Q^{(7)}(x)$ in Table IV gives the final quadratic factor as $(x^2+4.0488x+5.0287)$, with complex roots at $-2.024 \pm 0.965i$. The remainder, $R^{(7)}(x) \equiv x+0.0919$, gives the only real root at -0.092.

Example 3

We demonstrate the ability of the method to overcome the problem of non-convergence of the iteration by changing the p parameter value in the divisor. Consider the cubic given by

$$P(x) \equiv x^3 - 5x^2 + 9x - 9.$$

Using $\mathrm{d}P(x)/\mathrm{d}x$ as the initial divisor gives

$$Q^{(1)}(x) = \frac{x}{3} - \frac{5}{9} \quad \text{and} \quad R^{(1)}(x) = \frac{4x}{9} - 4.$$

However, choosing $p = 0$ gives a non-convergent (or at best a very slowly convergent) sequence of quotients and remainders $Q^{(i)}(x)$ and $R^{(i)}(x)$, $i = 2, 3, \ldots$.

This problem can be overcome by choosing a different value for p which necessarily matches the polynomial to its deflated approximant at this point. Such a value can be obtained from information gained about the nature of the polynomial or it can be systematically "guessed" (easily done by a computer program incorporating a simple subroutine for this purpose)—procedures which are common to all numerical methods. In this case, using $p = 3.5$ gives the convergent sequence of quotients listed in Table V. Using Eq. (7) with the coefficients given by $Q^{(9)}(x)$ in Table V gives a quadratic factor with complex roots at $1 \pm 1.4144i$, the associated remainder term $R^{(9)}(x)$ gives the remaining root at 3. The actual roots of $P(x)$ are 3 and $1 \pm \sqrt{2}i$, where it is seen that, to within the accuracy worked to, the method gives very good estimates.

Example 4

This example demonstrates how the method may be modified to overcome the "singular" case of a remainder polynomial not of degree one less than the divisor. Consider the polynomial

TABLE IV

	$Q^{(i)} \equiv ex+d$	
i	e	d
1	1.0000	1.3802
2	−4.7731	26.3155
3	0.0171	0.0705
4	0.1863	0.7553
5	0.1980	0.8015
6	0.1988	0.8049
7	0.1989	0.8051
8	0.1989	0.8051

TABLE V

	$Q^{(i)}(x) \equiv ex+d$	
i	e	d
2	2.25	16.8750
3	0.0140	0.0273
4	0.0958	0.1538
5	0.1145	0.1749
6	0.1194	0.1800
7	0.1207	0.1813
8	0.1211	0.1817
9	0.1212	0.1818
10	0.1212	0.1818

$$P(x) \equiv 4x^4 - x - 8.$$

Division of $P(x)$ by its derivative $16x^3 - 1$ gives a quotient $Q^{(1)}(x) = 0.25x^3$ and a remainder $R^{(1)}(x) = -0.75x - 8$. To use the iteration Eq. (5) in this case the divisor polynomial $(x-p)R^{(1)}$ must be of degree 3. To bring this divisor up to the correct degree it may be multiplied by an appropriate linear factor $(x-q)$, where q is a point at which the full and deflated polynomial will be matched. The rest of the iteration is carried out as before.

For this polynomial, choosing $p = 0$ gives a non-convergent iteration and so $p = 1$ was tried with $q = 1$ also, this gave the sequence of quotients in Table VI. Using Eq. (7) with the coefficients of $Q^{(9)}(x)$ in Table VI gives the quadratic factor with complex roots at $-0.0442 \pm 1.1900i$, and the remainder $R^{(9)}(x)$ gives the other two roots at 1.2326 and -1.1442, which are correct to within the accuracy worked to.

TABLE VI

	$Q^{(i)}(x) \equiv ex+d$	
i	e	d
2	−5.3333	0.0318
3	0.0105	0.0310
4	0.1707	0.4014
5	0.2476	0.4691
6	0.3216	0.4843
7	0.3762	0.4553
8	0.3977	0.4356
9	0.3990	0.4342
10	0.3990	0.4342

VI. Conclusions

A method for finding roots of polynomials with real coefficients has been given which is based upon extracting quadratic factors iteratively via a division algorithm. The method is simple to use and is very robust because of its facility to overcome a non-convergent iteration by variation of a single parameter p in the divisor factor $(x-p)$. It is particularly useful for coping with equal or nearly equal roots, which is a result of using the derivative of the polynomial as the initial divisor, thus ensuring that the deflated polynomial estimate (the remainder term) also contains one of these roots. Another advantage of the method is that only one parameter, p, needs to be "guessed" for the iteration process, unlike the methods of Lin and Bairstow where two parameters are required for the initial quadratic factor estimate.

Regarding the convergence of the method, it is seen that because the deflated estimate is essentially a multipoint Taylor approximant of the full polynomial, which matches function values only about distinct points in general, then the iteration, when convergent, will converge in a linear fashion. The link with the Newton–Raphson method suggests that quadratic convergence results only if function values and their first derivatives are matched at the roots of the divisors in the iteration—this is an area for further study. However, as with all numerical methods, the actual rate of convergence depends ultimately on the root distribution of the polynomial, and in this respect the author has found that the method compares very favourably with those of Lin and Bairstow over a wide range of examples.

Finally, it is expected that the technique will be a useful contribution to practical methods of polynomial root-finding, given its simplicity and easy computer implementation. In particular, it should appeal to control engineers, readily following their enthusiasm for Lin's method and their familarity with the Routh Array-type algorithm for polynomial division.

References

(1) F. B. Hildebrand, "Introduction to Numerical Analysis", 2nd edn., McGraw-Hill, New York, 1974.
(2) D. R. Towill, "Transfer Function Techniques for Control Engineers", Iliffe Books Ltd., London, 1970.
(3) A. C. Aitken, "On the theory of methods of factorizing polynomials by iterated division", *Proc. R. Soc.* Edinburgh, Vol. 64, pp. 326–335, 1952.
(4) C. F. Chen and M. M. Chen, "Performing Lin's method via Routh-type algorithms or Hurwitz-type determinants", *Proc. IEEE*, Vol. 68, pp. 1447–1449, 1980.
(5) T. N. Lucas, "Factor Division: a useful algorithm in model reduction", *IEE Proc. D, Control Theory and Applic.*, Vol. 130, pp. 362–364, 1983.
(6) P. Henrici, "Applied and Computational Complex Analysis", Vol. 1, Wiley, New York, 1974.
(7) R. C. Dorf, "Modern Control Systems", p. 167, Addison-Wesley, Reading, MA, 1967.

Appendix

Suppose it is required to divide the nth degree polynomial

$$P(x) \equiv a_0x^n + a_1x^{n-1} + \cdots + a_n$$

by the mth ($m < n$) degree polynomial

$$D(x) \equiv b_0 x^m + b_1 x^{m-1} + \cdots + b_m$$

from highest powers of x. This will give a quotient,

$$Q(x) \equiv c_0 x^{n-m} + c_1 x^{n-m-1} + \cdots + c_{n-m}$$

and remainder,

$$R(x) \equiv d_0 x^{m-1} + d_1 x^{m-2} + \cdots + d_{m-1}.$$

The coefficients c_i and d_j ($i = 0, 1, \ldots, n-m$; $j = 0, 1, \ldots, m-1$) may be conveniently calculated from the Routh-type synthetic division array (**4, 5**) which now follows:

a_0	a_1	.....		a_{n-1}	a_n
b_0	b_1	.....	b_m	$0 \ldots$ 0	0
h_{11}	h_{12}	h_{13}	.....	$h_{1,n}$	
h_{21}	h_{22}	h_{23}	....	$h_{2,n-1}$	
⋮	⋮	⋮			
$h_{n-m,1}$	.....	....	$h_{n-m,m+1}$		
d_0	d_1	...	d_{m-1}		

where

$$h_{i,j} = h_{i-1,j+1} - c_{i-1} b_j \quad i = 1, 2, \ldots, n-m, \quad j = 1, 2, \ldots, n$$

$$c_i = h_{i,1}/b_0 \quad i = 0, 1, 2, \ldots, n-m, \quad b_i = 0 \quad i > m,$$

$$h_{0,j} = a_{j-1} \quad j = 2, 3, \ldots, n+1$$

and

$$d_{i-1} = h_{n-m,i+1} - c_{n-m} b_i \quad i = 1, 2, \ldots, m.$$

For instance, using the example of division in Section II gives the array,

3		−2	4	5	−2
1		−4	5	−2	
10		−11	11	−2	
29		−39	18		

which gives the quotient from the first column as $3x + 10$ and the remainder from the last row as $29x^2 - 39x + 18$.