On Newton's method with a class of rational functions

David Benjamin Clegg (*)

ABSTRACT

The classical Newton's method for determining a real factor of a polynomial is shown to be one member of a family of similar algorithms found by considering the roots of a class of rational functions. These algorithms are variants of the Birge Vieta method for solving polynomial equations. Methods for selecting an appropriate member of the family for a general problem are compared. Third order possibilities are shown to exist and results for a particular case are reported.

1. INTRODUCTION

Methods for solving polynomial equations e.g. Grant and Hitchins [6] which always converge, are to be preferred to those methods which depend for success upon a good initial approximation. However, at the same time, the relative rates of convergence of methods can be an important practical consideration, and the computational efficiency of a method depends upon the number of function evaluations (and derivative calculations in methods such as Newton-Raphson). Methods for solving non-linear equations are well documented in the classic works of Traub [14] and Ostrowski [10] and a purposeful review has been given by Jarratt [8].

Central to the discussion of a method are the concepts of order, p, and asymptotic error constant, C, and in the neighbourhood of a root the speed of convergence will depend upon both these, being most rapid when p is large and C is small. It is likely that C and hence the rate of convergence can vary with the choice of iteration function. In this respect, Brodlie [5] has shown that Bairstow's method [2] is one member of a family of algorithms for determining a quadratic factor of a polynomial and he demonstrated by experimentation that superior rates of convergence were possible through the purposeful choice of error terms. As Bairstow's method is an extension of Newton's method for finding a simple real root or real linear factor of a polynomial with real coefficients, Brodlie [5] suggested that a similar approach may lead to a superior algorithm for the calculation of real linear factors and hence real roots.

Consider, therefore

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0, \qquad (1.1)$$

$$a_n \neq 0$$

a real polynomial of degree n in x. A simple real root, a, of $f(x) = 0$, is found by determining the linear factor $(x - a)$ such that

$$f(x) = (x-a)(b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \ldots + b_1 x + b_0) \qquad (1.2)$$

On equating coefficients of x through (1.1) and (1.2), $(n+1)$ equations are obtained in the n unknowns $b_{n-1}, b_{n-2}, \ldots, b_1, b_0$ and the system of equations is therefore overdetermined. The equations are consistent if a is a root and otherwise they can be made so by the addition of an arbitrary remainder term $u_r x^r$, $0 \leq r \leq n$, to the right hand side of (1.2).

Thus for any value of a

$$f(x) = (x-a)(b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \ldots b_1 x + b_0) + u_r x^r \qquad (1.3)$$

With exact computation

$$u_r(a) = \frac{f(a)}{a^r}, \quad a \neq 0. \qquad (1.4)$$

Hence a root of $f(x)$ will also be a root of the rational functions $u_r(x)$, $r \in \{0, 1, 2, \ldots, n\}$. These $(n+1)$ rational functions provide a set of alternative functions for solving the polynomial $f(x) = 0$, through any iteration formula. It is reasonable to suppose that for a particular method, the asymptotic error constant associated with the function $u_r(x)$, will depend upon r. In this paper this effect is studied in respect of Newton's method, the classical Newton-Raphson formula arising when $r = 0$.

2. ANALYSIS

Suppose that $f(x)$ has a zero, a, of multiplicity p, then

$$f(x) = (x-a)^p g(x), \qquad g(a) \neq 0 \qquad (2.1)$$

(*) D. B. Clegg, Department of Mathematics, Liverpool Polytechnic, Liverpool, England.

The Newton-Raphson method

$$x_{j+1} = x_j - \frac{f(x_j)}{f'(x_j)}\ ; \quad x_0 \text{ given, } j = 0,1,2,\ldots \tag{2.2}$$

can be modified [13] so that it still has quadratic convergence for a root of multiplicity p by writing

$$x_{j+1} = x_j - p\,\frac{f(x_j)}{f'(x_j)}\,. \tag{2.3}$$

Hence substituting (1.4) for function f and following Bodewig [4], the general formula may be set as

$$x_{j+1} = x_j - a\,\frac{u_r(x_j)}{u_r'(x_j)} \tag{2.4}$$

i.e.

$$x_{j+1} = x_j - a\,\frac{f(x_j)}{f'(x_j) - \frac{r}{x_j}\,f(x_j)}\,, \quad x_j \neq 0 \tag{2.5}$$

with a, a real constant to be determined after examining the character of convergence of the sequence $x_0, x_1, x_2, \ldots$ which is assumed to tend to a root of multiplicity p. Assuming necessary conditions on differentiability, $g(x_j)$ in (2.1) can be expanded about $x = a$ through

$$g(x_j) = g[a + (x_j - a)] = g(a) + \epsilon_j g'(a) + \frac{\epsilon_j^2}{2}\,g''(a) + \ldots \tag{2.6}$$

with $\epsilon_j = x_j - a$ (2.7)

Also $$g'(x_j) = g'(a) + \epsilon_j g''(a) + \frac{\epsilon_j^2}{2}\,g'''(a) + \ldots \tag{2.8}$$

Substituting (2.1) into (2.5) applying (2.6) and (2.8), assuming that ϵ_j is sufficiently small and subtracting a from both sides of (2.5) produces the equation

$$\epsilon_{j+1} = (1 - \tfrac{a}{p})\,\epsilon_j + \frac{a}{p^2}\left[\frac{g'(a)}{g(a)} - \frac{r}{a}\right]\epsilon_j^2 + \frac{a}{p}\left\{\left[\frac{g'(a)}{g(a)} + \frac{1}{a}\right]\left[\frac{1}{a}\left(1 - \frac{r}{p}\right) + \frac{g'(a)}{g(a)}\left(1 + \frac{1}{p}\right)\right] + \frac{g'(a)}{g(a)}\left(\frac{1-r}{ap}\right) - \frac{1}{a^2}\left[(p-r)\frac{g'(a)}{g(a)} + \left(1 - \frac{r}{p}\right)^2\right]\right\}\epsilon_j^3 + \ldots \tag{2.9}$$

Hence we have the following results :

i) If a = p, then in the neighbourhood of a root of multiplicity p, formula 2.5 will yield a sequence which converges quadratically to the root (if it converges at all).

ii) If a = p and $$\frac{g'(a)}{g(a)} - \frac{r}{a} = 0 \tag{2.10}$$

the method will be at least third order.

Using (2.1) it is readily shown that

$$g(a) = \frac{f^{(p)}(a)}{p!} \tag{2.11}$$

$$g'(a) = \frac{1}{(p+1)!}\,f^{(p+1)}(a) \tag{2.12}$$

where $f^{(p)}(a) = \dfrac{d^p f}{dx^p}(a)$.

Thus for cubic convergence it is necessary that

$$\frac{1}{(p+1)}\,\frac{f^{(p+1)}(a)}{f^{(p)}(a)} - \frac{r}{a} = 0. \tag{2.13}$$

If we now consider simple roots, p = 1 = a then the method is third order if

$$\frac{f''(a)}{2f'(a)} - \frac{r}{a} = 0. \tag{2.14}$$

Additionally when the method is second order, a convergent sequence will have asymptotic error constant C given by

$$C = \frac{f''(a)}{2f'(a)} - \frac{r}{a}\,. \tag{2.15}$$

Hence the constant is seen to be dependent upon r and thus an optimum value of r may be possible to minimise the modulus of C.

iii) When (2.10) is satisfied, the third order term in (2.9) reduces to

$$\frac{a}{a^2 p}\left[r^2(1+p) + 2r(p-1) + p - 1\right]\epsilon_j^3 \tag{2.16}$$

This term only equals zero if

$$r = \frac{-(p-1) \pm \sqrt{2-2p}}{(p+1)}\,.$$

When p = 1, then r = 0 and the conditions for the standard Newton-Raphson method apply.
If p > 1 the solutions will be complex and hence as r is an integer, the formula (2.5) can be third order at best for simple roots.

Let us now turn to the possibilities of computational advantages to be gained using (2.5) with a = 1, r ≠ 0, in preference to the standard Newton-Raphson formula. The fundamental problem is the selection of r. Peters and Wilkinson [11] have commented that there is no reason to believe that there is any advantage to be gained in using $u_r(a)$, $1 \leq r \leq n$ over $u_0(a)$, provided that a is calculated with such precision as one is prepared to employ, and that for subsequent roots, composite deflation is made. This assertion appears to neglect the possible advantages to be associated with a minimum absolute value of the asymptotic error constant for the "best" value of r. However, the selection of r is not straightforward as initially a is not known and at best only estimations of C can be obtained with a replaced by x_j, $j \in \{0,1,2,\ldots\}$. $u_r(x_j)$ and $u_r'(x_j)$ can be determined through a combination of forward and backward division of the polynomial by a linear factor as in composite deflation by Peters and Wilkinson [12]. Hence these are a family of methods which are variants of the Birge Vieta method [3], i.e. Newton iteration to find a root of a polynomial with f and f′ calculated via the Horner scheme. The additional computation associated with the selection of r,

may possibly counterbalance any advantage of superior convergence. Results with numerical experiments which appear later in the paper, exhibit this tendency.

3. CALCULATION OF u_r AND u_r'

When dividing $f(x)$ by $(x-a)$ with forward division, the coefficients $b_{n-1}, b_{n-2}, \ldots, b_1, b_0$ in (1.2) are found through the equations

$$b_{n-1} = a_n$$

$$b_i = a_{i+1} + a b_{i+1}, \qquad i = n-2, n-3, \ldots, 0, -1$$

with $b_{-1} = u_0$

With full backward division of $f(x)$ by $(x-a)$ we would have

$$f(x) = u_n x^n + (x-a)(c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \ldots + c_1 x + c_0) \cdot \tag{3.2}$$

The coefficients being calculated through

$$\begin{cases} c_0 = + \dfrac{a_0}{(-a)} \\ c_i = \dfrac{(a_i - c_{i-1})}{(-a)} & i = 1, 2, 3, \ldots, n-1 \\ c_n = a_n - c_{n-1} = u_n \end{cases} \tag{3.3}$$

The remainder term $u_r x^r$ can generally be found through a mixture of backward and forward division from

$$f(x) = (x-a)(b_{n-1}x^{n-1} + \ldots + b_r x^r) + u_r x^r + (x-a)(c_{r-1}x^{r-1} + c_{r-2}x^{r-2} + \ldots + c_1 x + c_0) \tag{3.4}$$

The resulting equations found by equating coefficients of x are

$$\begin{cases} b_{n-1} = a_n \\ b_i = a_{i+1} + a b_{i+1} & i = n-2, \ldots, r+1, r, r-1 \\ c_0 = \dfrac{a_0}{(-a)} \\ c_i = \dfrac{(a_i - c_{i-1})}{(-a)} & i = 1, 2, 3, \ldots, r-1 \\ \text{and } u_r = b_{r-1} - c_{r-1} & 0 < r < n. \end{cases} \tag{3.5}$$

In order to calculate u_r' the above equations are differentiated with respect to a, and without loss of generality it can be assumed that $a_n = 1$ always.
Differentiating (3.5) with respect to a and writing $\dfrac{db_i}{da} = d_i$, $\dfrac{dc_i}{da} = e_i$ we have

$$d_{n-1} = 0$$

$$d_i = b_{i+1} + a d_{i+1} \qquad i = n-2, n-1, \ldots, r, r-1 \tag{3.6}$$

$$e_0 = \frac{a_0}{a^2} = \frac{c_0}{(-a)}$$

$$e_i = \frac{c_i - e_{i-1}}{(-a)} \qquad i = 1, 2, 3, \ldots, r-1$$

$$u_r' = d_{r-1} - e_{r-1} \qquad 0 < r < n$$

Also u_0' is found through $u_0' = d_{-1}$

and u_n' through $u_n' = -e_{n-1}$ (3.7)

4. SELECTION OF r

In considering the choice of r, it should be noted that r need not be fixed initially but may be varied with each value of x_j, $j = 0, 1, 2, \ldots$ Clearly additional computation will be necessary in order that the "best" value of r is chosen and in any practically useful algorithm, the computational effort must be accounted for. Initially we neglect this problem and consider the options upon the selection of r.
Rearranging (3.4) we have :

$$(x-a)(b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \ldots + b_r x^r + c_{r-1}x^{r-1} + c_{r-2}x^{r-2} + \ldots + c_1 x + c_0) = a_n x^n + a_{n-1}x^{n-1} + \ldots + (a_r - u_r)x^r + \ldots + a_1 x + a_0 \tag{4.1}$$

The right hand side of (4.1) may be regarded as a perturbation of the original polynomial. Hence thinking of u_r as a perturbation on a_r, r could be selected in order that $\left|\dfrac{u_r}{a_r}\right|$, $0 \leq r \leq n$, is a minimum.
This is the identical requirement specified by Peters and Wilkinson [10] for satisfactory composite deflation. When a is obtained to machine accuracy say, there is the possibility of immediately deflating the polynomial using the value of r as selected, to also calculate the deflated polynomial.
Another possible variant for the choice of r is to select r such that $|u_r|$, $0 \leq r \leq n$ is a minimum. Intuitively one would expect this to be an inferior choice to the one described above, as no attempt is made to relate u_r with the associated perturbed polynomial. For ill-conditioned polynomials this could be a dangerous criterion to apply.
A third variant on the choice of r is suggested through the coefficient of ϵ_j^2 in (2.9). In other words we use a criterion which seeks the minimum modulus of the asymptotic error constant of the formula (2.5), remembering that $a = p = 1$.

For the standard Newton-Raphson method, the asymptotic error constant is given by $\frac{f''(a)}{2f'(a)}$. Hence improvement in the rate of convergence can be anticipated whenever

$$\left|\frac{f''(a)}{2f'(a)} - \frac{r}{a}\right| < \left|\frac{f''(a)}{2f'(a)}\right| ; \quad 1 \leq r \leq n. \tag{4.2}$$

If $\frac{f''(a)}{2f'(a)}$ equals zero, no improvement will be possible. For functions satisfying such a condition, the standard Newton-Raphson method will yield a third order solution.

Two other cases need considering, depending upon the sign of $\frac{f''(a)}{f'(a)}$.

i) $\frac{f''(a)}{f'(a)} > 0.$

For this condition, (4.2) rearranges to

$$0 < \frac{r}{a} < \frac{f''(a)}{f'(a)}, \quad 1 \leq r \leq n. \tag{4.3}$$

Hence a value of $r > 0$ is only possible when a is a positive zero of the polynomial (1.1)

ii) $\frac{f''(a)}{f'(a)} < 0.$

For which condition (4.2) rearranges to

$$\frac{f''(a)}{f'(a)} < \frac{r}{a} < 0 \tag{4.4}$$

Consequently a value of $r > 0$ can only possibly be preferred if a is a negative zero of the polynomial.

Given that the conditions (4.3) and (4.4) are possible, r should be selected so as to minimize the left hand side of (4.2).

The practical difficulty with each of the proposed variants, is that a, a root, needs to be known if r is to be correctly determined. Initially a is approximated by a guess value x_0 and this value may be applied to indicate the optimum value of r. For the first two variants, the values $u_r(x_0)$, $0 \leq r \leq n$ are calculated through equations (3.1), (3.3) and (3.5) as appropriate; with x_0 replacing a. For the third variant, we require to evaluate $u_0'(x_0)$ and $u_0''(x_0)$. $u_0'(x_0)$ is readily found with d_{-1} calculated from (3.6) and x_0 replacing a. With an additional synthetic division, the second derivative

$$u_0''(x_0) = 2f_{-1}, \tag{4.5}$$

where $f_{n-3} = d_{n-2}$ (4.6)

$$f_i = e_{i+1} + af_{i+1}, \quad i = n-4, \ldots, 0, -1 .$$

Now consider the computational effort required to calculate $u_r(x_j)$. Once the value r is selected, the total number of arithmetic operations to evaluate the next iterate from equation (2.4), will be identical to that necessary for the classical Newton-Raphson formula. Some relevant operation counts are listed in the table below :

TABLE 1. Operation counts

Function calculated	Multiplications	Division	Additions	Subtractions
$u_r(a)$, $0 \leq r \leq n$	n−r	r	n−r	r
$u_r'(a)$, $0 \leq r \leq n$	n−r	r	n−r	r
All $u_r(a)$; $r=0,1,2,\ldots,n$	n	n	n	n−1
All $\frac{u_r(a)}{a_r}$; $r=0,1,2,\ldots,n$, $a_r \neq 0$	n	2n	n	n−1

The computational effort may be defined through a combination of the number of operations and the execution time associated with each. These times vary through computer installations but for the sake of simplicity, if approximately equal computation times are assumed for multiplication and division and different but equal times for addition and subtraction, the table values show that twice the computational effort will be required to calculate all values of $u_r(a)$, $r = 0, 1, \ldots, n$, compared with that needed for a selected value of r. Also approximately three times the effort is required for calculating $(\frac{u_r(a)}{a_r})$, $r = 0, 1, \ldots, n$. Finally for large n, the calculation of $\frac{f''(a)}{2f'(a)}$ requires approximately one and a half times the effort for calculating $\frac{f(a)}{f'(a)}$.

Consequently in order that any of the proposed variants should be preferred to the standard Newton-Raphson method the overhead of the initial computation to select r must be compensated by a corresponding decrease in the required numbers of iterations. Additionally the performance determined by the percentage of successful calculations must be at least as good as that obtained through the use of the standard methods.

5. ERROR BOUNDS FOR $u_r(a)$

The stopping criterion associated with Adams [1] can be applied to determine $u_r(a)$ to the limit of machine accuracy. Peters and Wilkinson [12] have given a detailed account for calculating a realistic error bound for $f(a)$ i.e. $u_0(a)$. Their analysis can be extended to compute a running error bound on the calculation of $u_r(a)$. With forwards division of $f(x)$ by $(x-a)$, a convenient form for the bound on $u_0(a)$ has been given by Kahan [9]. For the sake of completeness and understanding a summary of the assumptions and results already derived are given.

The exact process for forward and backward division

of f(x) by $(x-a)$ are given by equations (3.1), (3.3) and (3.5). Following Wilkinson [15] the following assumptions are made about floating point arithmetic operations for binary arithmetic with a t digit mantissa.

$$\begin{aligned} fl(x \pm y) &= (x \pm y)/(1+\epsilon), & |\epsilon| &< 2^{-t} \\ fl(x * y) &= (x * y)(1+\epsilon), & |\epsilon| &< 2^{-t} \\ fl(x/y) &= x(1+\epsilon)/y, & |\epsilon| &< 2^{-t} \end{aligned} \tag{5.1}$$

The computational processes used are given by the equations :

$$s_{n-1} = a_n, \quad s_i = fl(a s_{i+1} + a_{i+1}); \quad i = n-2, n-3, \ldots, 0, -1 \tag{5.2}$$

$$\begin{cases} t_0 = fl[a_0/(-a)], \quad t_i = fl[(a_i - t_{i-1})/(-a)]; \\ \qquad\qquad i = 1, 2, \ldots, n-1 \\ t_n = fl(a_n - t_{n-1}) \end{cases} \tag{5.3}$$

$$\begin{cases} v_0 = a_{-1} \\ v_i = fl(s_{i-1} - t_{i-1}); \quad 0 < i < n \\ v_n = t_n \ . \end{cases} \tag{5.4}$$

Now we define the errors in s_i, t_i and v_i respectively through

$$\begin{cases} s_i = b_i + g_i; & i = n-1, n-2, \ldots, 0, -1 \\ t_i = c_i + h_i; & i = 0, 1, 2, \ldots, n \\ v_i = u_i + k_i; & i = 0, 1, 2, \ldots, n. \end{cases} \tag{5.5}$$

Combining (5.2) with (5.1) we find

$$s_i = [a s_{i+1}(1+\epsilon_i) + a_{i+1}]/(1+\eta_i); \quad i = n-1, n-2, \ldots, 0, -1 \tag{5.6}$$

$$|\eta_i|, |\epsilon_i| \leq 2^{-t} \ .$$

Hence $s_i = a s_{i+1}(1+\epsilon_i) + a_{i+1} - s_i \eta_i$. (5.7)

Substituting for s_i from (5.5) yields

$$g_i = g_{i+1} + a s_{i+1} \epsilon_i - s_i \eta_i, \tag{5.8}$$

from which it is readily shown that

$$|g_{n-r}| \leq \left(|a^{r-1} s_{n-1}| + 2|a^{r-2} s_{n-2}| + \ldots + 2|a s_{n-r+1}| + |s_{n-r}|\right) 2^{-t} = G_{n-r} 2^{-t}, \quad r = 2, 3, 4, \ldots, n, n+1 \tag{5.9}$$

Using backwards division of f(x) by $(x-a)$ and combining (5.3) with (5.1) we find :

$$t_0 = -\frac{a_0}{a}(1+\gamma_0), \qquad |\gamma_0| < 2^{-t} \tag{5.10}$$

Substituting for h_0 from (5.5) gives

$$h_0 = \frac{\gamma_0}{1+\gamma_0} t_0 \tag{5.11}$$

Also from (5.3) and (5.1) we have

$$t_i = \frac{(a_i - t_{i-1})}{(1+\xi_i)} \frac{(1+\delta_i)}{(-a)}; \quad i = 1, 2, \ldots, n-1 \qquad |\xi_i|, |\delta_i| \leq 2^{-t} \tag{5.12}$$

Proceeding as above with substituting for t_i in terms of h_i, then

$$h_i = \frac{h_{i-1}}{a} + t_i \frac{(\delta_i - \xi_i)}{(1+\delta_i)} \tag{5.13}$$

from which it can be deduced that

$$|h_r| \leq \left[\left|\frac{t_0}{a^r}\right| + 2\left|\frac{t_1}{a^{r-1}}\right| + \ldots + 2|t_r|\right] \frac{2^{-t}}{1-2^{-t}}$$

$$= H_r \frac{2^{-t}}{1-2^{-t}} \qquad r = 1, 2, \ldots, n-1 \tag{5.14}$$

Finally

$$t_n = \frac{a_n - t_{n-1}}{1+\xi_n} \ . \tag{5.15}$$

Hence substituting $t_n = c_n + h_n$,

$$h_n = -h_{n-1} - \xi_n t_n \tag{5.16}$$

Consequently

$$|h_n| \leq |h_{n-1}| + |t_n| 2^{-t} \leq [H_{n-1} + |t_n|(1-2^{-t})] \frac{2^{-t}}{1-2^{-t}}$$

$$= H_n \frac{2^{-t}}{1-2^{-t}} \ . \tag{5.17}$$

Following Kahan [9] the most convenient form for calculating G_r and H_r, $r = n-1, n-2, \ldots, 0, -1$ is through

$$\begin{cases} K_{n-1} = \frac{1}{2}|s_{n-1}| \\ K_r = |a| K_{r+1} + |s_r|; & r = n-2, n-3, \ldots, 0, -1 \\ G_r = 2K_r - |s_r|; & r = n-1, n-2, \ldots, 0, -1 \end{cases} \tag{5.18}$$

$$\begin{cases} L_0 = \frac{1}{2}\left|\frac{t_0}{a}\right|; \\ L_r = \frac{L_{r-1}}{|a|} + |t_r|; & r = 1, 2, \ldots, n-1 \\ H_r = 2L_r; & r = 1, 2, 3, \ldots, n-1 \\ H_n = H_{n-1} + |t_n|(1-2^{-t}). \end{cases} \tag{5.19}$$

Finally, we obtain a bound on the computed value for u_r, $0 < r < n$. By (5.4)

$$v_r = fl(s_{r-1} - t_{r-1}) \tag{5.20}$$

$$= (s_{r-1} - t_{r-1})/(1+\theta_r), \quad |\theta_r| \leq 2^{-t} \tag{5.21}$$

Hence $v_r = s_{r-1} - t_{r-1} - v_r \theta_r$. (5.22)

Substituting $v_r = u_r + k_r$, and also for s_{r-1} and t_{r-1} from (5.5)

$$k_r = g_{r-1} - h_{r-1} - v_r \theta_r \tag{5.23}$$

$$|k_r| \leq |g_{r-1}| + |h_{r-1}| + |v_r| 2^{-t} \tag{5.24}$$

$$|k_r| \leq G_{r-1} + H_{r-1} + |v_r|\,2^{-t};\ 0 < r < n \qquad (5.25)$$

Thus the error bounds on the computed values of $u_r(a)$ are given by

$$\begin{cases} |u_0 - v_0| \leq G_{-1} \\ |u_r - v_r| \leq G_{r-1} + H_{r-1} + |v_r|\,2^{-t} \\ |u_n - v_n| \leq H_{n-1} \cdot \qquad r = 1,2,\ldots, n-t \end{cases} \qquad (5.26)$$

For the practical computation of H_{n-1} and G_{-1}, we take $a \triangleq x_j$, and hence obtain bounds on $u_n(x_j)$ and $u_0(x_j)$.

It should be noted that the sequence $\{G_r;\ r = n-2,\ldots, 0, -1\}$ is non decreasing if $|a| \geq \frac{1}{2}$, and the sequence $\{H_r;\ r = 0, 1,\ldots, n-1\}$ may be a decreasing sequence if $|a| \geq 2$. Thus if $|a| \geq 2$, the bound on $u_0(a)$ i.e. G_{-1}, is calculated through a non decreasing sequence whilst that for $u_n(a)$ i.e. H_{n-1}, is found using a possibly decreasing sequence. Under such conditions, the bound on $u_n(a)$ may be much smaller than that of $u_0(a)$ and hence when the convergence criterion is satisfied, a better estimate for a may be possible using full backward division. The converse of this applies when $|a| \leq \frac{1}{2}$, and for $\frac{1}{2} < |a| < 2$, the situation is always indefinite. As no firm conclusion can be drawn from these observations, common sense would suggest that an extra iteration be made after the relevant convergence criterion has been satisfied for the computed value of $u_r(a)$, $r \in \{0, 1,\ldots, n\}$.

6. NUMERICAL RESULTS

In this section, the relative performance of the three methods described in section 4 are compared with each other and also against the standard Newton-Raphson method. For ease of reference the following descriptions will be used :

METHOD 1 - Classical Newton-Raphson method, $r = 0$.

METHOD 2 - r is selected so that $\left|\frac{u_r(x_0)}{a_r}\right|$ is a minimum.

METHOD 3 - r is selected so that $|u_r(x_0)|$ is a minimum.

METHOD 4 - r is selected so that $\left|\frac{1}{2}\frac{f''(x_0)}{f'(x_0)} - \frac{r}{x_0}\right|$ is a minimum.

In view of the computation overheads in selecting r, with methods 2, 3, and 4, algorithms used selected r on the basis of the initial guess value x_0, and r as fixed thereafter. A method was assumed to have failed if convergence was not achieved in 12 iterations, all calculations being made to the limit of machine accuracy on $u_r(x_j)$.

For experimentation, twenty eight polynomials were used. Twenty seven of these are a subset of the polynomials used by Henrici and Watkins [7], the selection being determined by the existence of real and fairly well separated roots, i.e. polynomials 1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32. Polynomial twenty eight was found in the paper by Peters and Wilkinson [12], this being chosen because of the wide separation of the roots and the reported large differences in deflated polynomials through combinations of forward and backward division of the polynomial.

All methods were used to determine roots of the polynomials using four different initial guesses, these being ± 5 % and ± 10 % from the known roots. In total seventy one test cases were attempted with the real roots of the identified polynomials. The results are summarized in the following table where information is found on :

i) the number of problems successfully solved by the method used.
ii) the number of problems for which the method converged to a root not being the closest to the initial guess, these figures appear in parentheses.
iii) the average number of iterations taken to achieve convergence.

TABLE 2. Number of iterations for solved problems

Error in initial guess	Number of problems solved				Average number of iterations			
	METHOD				METHOD			
	1	2	3	4	1	2	3	4
+5%	71	71	71	71	4.04	3.77	4.48	3.14
-5%	70	71	71	70	4.26	3.49	3.87	3.24
+10%	70	71	66	71	4.59	4.23	4.73	3.55
-10%	62(5)	71	71	61(5)	4.60	4.03	4.41	3.54

The results show that :

i) Method 2 has the highest likelihood of convergence from a given initial guess; 100 % success being achieved with the chosen test data.
ii) When successful, method 4 has the lowest average number of iterations, this being one less than for method 1, i.e. the standard Newton-Raphson method.
iii) Method 4 is not very reliable when used with poor initial guesses, (when r was selected at each stage, 100 % success was obtained for all the chosen test data).
iv) Method 3 is the least preferred, exhibiting the highest number of iterations on average.

In order to assess any advantage in the computational effort required by the various methods, it is necessary to evaluate the overheads associated with the selection

of r. Defining a Horner [10] as the number of multiplications and divisions required in the computation of both $u_r(x_0)$ and $u_r'(x_0)$, $0 \leqslant r \leqslant n$, then the number of non recoverable Horner units for each method, are as follows :

Method 1 - Nil
Method 2 - 1
Method 3 - 0.5
Method 4 - between 0.5 and 1.5, say 1 on average.

Using the data of table 2, the average number of Horners associated with the test data are shown in table 3.

TABLE 3. Average number of Horners required for each method

Error in initial guess	Method 1	Method 2	Method 3	Method 4
+ 5 %	4.04	4.77	4.98	4.14
- 5 %	4.26	4.49	4.37	4.24
+ 10 %	4.59	5.23	5.23	4.55
- 10 %	4.60	5.03	4.91	4.54

Hence in general the three proposed variants do not offer any advantage in saving computational effort. This confirms the general statement made by Peters and Wilkinson [11] and is in contrast to the findings of Brodlie [5]. In specific cases where r can be prechosen, there are computational advantages, particularly for close approximations to the root. Such possibilities are indicated through a particular polynomial quoted by Peters & Wilkinson [12] i.e.

$$f(x) = x^3 + 0.981318 \times 10^4 x^2 + 0.857108 \times 10^4 x + 0.781736 = 0 \qquad (6.1)$$

with roots

$x_1 = -0.912157 \times 10^{-4}$, $x_2 = -0.873412$,

$x_3 = -0.981231 \times 10^4$.

The table below shows the number of iterations required by the four methods used.

TABLE 4. (All results were calculated on an ICL 1903A)

Error in initial guess	Calculation of x_1				Calculation of x_2				Calculation of x_3			
	METHOD				METHOD				METHOD			
	1	2	3	4	1	2	3	4	1	2	3	4
+5%	2	2	2	2	3	3	3	2	4	4	4	2
- 5 %	2	2	2	2	4	2	4	2	4	2	4	2
+ 10%	2	2	2	2	4	4	4	2	4	4	4	2
- 10%	2	2	2	2	4	2	4	2	5	2	4	2

Finally as method 4 offers the possibility of becoming a third order method when (2.15) becomes zero, it is reasonable to suppose that there are classes of polynomials for which the method will have considerable advantages over the standard method and also methods 2 and 3. Only one of the test polynomials used was found to satisfy the required condition (number 19, reference [7]). This problem is further examined in the next section.

7. THIRD ORDER SOLUTIONS

The analysis of section 2 showed that method 4 becomes third order if

$$\frac{f''(a)}{2f'(a)} - \frac{r}{a} = 0, \quad 0 \leqslant r \leqslant n. \qquad (7.1)$$

Classes of polynomials which will have third order solutions, must satisfy (7.1), together with the conditions $f(a) = 0$, $f'(a) \neq 0$ (if roots are simple).

By writing $f(x) = (x-a)\,h(x)$ (7.2)

with $h(x)$ a polynomial of degree $(n-1)$ in x, it is readily deduced that

$$\frac{h'(a)}{h(a)} = \frac{r}{a} \qquad (7.3)$$

where $h(a) = f'(a) \neq 0$.

With $h(x)$ further expressed as

$$h(x) = (x-a)\,s(x) + \beta \qquad (7.4)$$

and $s(x)$ a polynomial of degree $(n-2)$ in x, $\beta \neq 0$, it can be deduced that

$$h(x) = (x-a)\,s(x) + \frac{a\,s(a)}{r} \qquad (7.5)$$

$$\therefore f(x) = (x-a)\left[(x-a)s(x) + \frac{a\,s(a)}{r}\right], \qquad (7.6)$$

$$1 \leqslant r \leqslant n$$

This form for a polynomial is clearly very restrictive and can at best only be applicable to special cases. However, a class of polynomials can be obtained by treating the differential condition (7.1) which is true for a a root of $f(x)$, as a differential equation. Solving (7.1) yields

$$f(x) = Ax^{2r+1} + B \qquad (7.7)$$

where A and B are constants. Without loss of generality (7.7) can be written in the form

$$f(x) = x^{2r+1} + a_0 \qquad (7.8)$$

with a_0 a real constant, other than zero. These polynomials have one real root with sign opposite to that of a_0. The third order property is simply illustrated through

$$u_r(x) = x^{r+1} + \frac{a_0}{x^r} \qquad (7.9)$$

and the iteration formula becomes

$$x_{j+1} = \frac{r x_j^{2r+1} - (r+1)\,a_0}{(r+1)x_j^{2r+1} - r a_0}\, x_j \qquad (7.10)$$

The standard Newton-Raphson formula is

$$x_{j+1} = \frac{1}{(2r+1)} \left(2r x_j - \frac{a_0}{x_j^{2r}}\right)$$

Comparative results are shown below for the case $r = 2$, $a_0 = -2$, beginning in each case with $x_0 = 1$.

Newton-Raphson Method

j	x_j
1	1.2
2	1.152 901 234 507 9
3	1.183 346 933 780 931
4	1.150 668 840 675 584
5	1.148 705 092 244 385
6	1.148 698 355 076 062
7	1.148 698 354 997 034
8	1.148 698 354 997 034

New Method

j	x_j
1	1.142 857 142 857 142
2	1.148 698 050 614 295
3	1.148 698 354 997 033
4	1.148 698 354 997 033

8. CONCLUSIONS

Investigations with a class of rational functions for solving polynomial equations and Newton's method have shown that in general there is little advantage in using a chosen rational function unless the asymptotic behaviour of the iteration method can be predicted in advance. Newton's method which is a second order method has been shown to have third order convergence properties for the equation

$$\frac{x^{2r+1} + a_0}{x^r} = 0 \qquad r = 0, 1, 2, \ldots \qquad (8.1)$$

Finally it should be noted that the same computational possibilities exist with other methods for both real and complex roots. These possibilities remain to be explored.

9. REFERENCES

1. ADAMS D. A. : "A stopping criterion for polynomial root finding", Comm. ACM 10, 1967, pp. 655-658.
2. BAIRSTOW L. : "Investigation relating to the stability of the aeroplane", Reports and Memoranda 154, Advisory Committee for Aeronautics, 1914, pp. 51-64.
3. BALFOUR M., McTERNAN A. J. : *The numerical solution of equations*, 1st ed., Heinemann, London, 1967.
4. BODEWIG E. : "On types of convergence and on the behaviour of approximation in neighbourhood of a multiple root of an equation", Quart. Appl. Math. 7, 1948, pp. 325-333.
5. BRODLIE K. W. : "On Bairstow's method for the solution of polynomial equations", Maths. Comp. 29, 1975, pp. 816-826.
6. GRANT J. A., HITCHINS G. D. : "An always convergent minimisation technique for the solution of polynomial equations", J. Inst. Maths. Appl. 8, 1971, pp. 122-129.
7. HENRICI P., WATKINS B. O. : "Finding zeroes of a polynomial by the QD algorithm", Comm. ACM. 8, 1965, pp. 570-574.
8. JARRATT P. : "A review of methods for solving nonlinear algebraic equations in one variable", in : *Numerical methods for nonlinear algebraic equations* (P. Rabinowitz, ed.), Gordon & Breach, London, 1971, pp. 1-26.
9. KAHAN W., FARKAS I. : "Algorithm 168 and Algorithm 169", Comm. ACM. 6, 1963, p. 165.
10. OSTROWSKI A. M. : *Solution of equations in Euclidean Banach spaces*, 1st ed., Academic Press, New York, 1973.
11. PETERS G., WILKINSON J. H. : "Practical problems arising in the solution of polynomial equations", J. Inst. Maths. Appl. 8, 1971, pp. 16-35.
12. PETERS G., WILKINSON J. H. : "On an algorithm for polynomial deflation by Broyden and Ford" : NPL. Report NAC 55.
13. RALSTON A., RABINOWITZ P. : *A first course in numerical analysis*, 2nd ed., McGraw Hill, New York, 1978, p. 354.
14. TRAUB J. F. : *Iterative methods for the solution of equations*, 1st ed., Prentice Hall, Englewood Cliffs, 1964.
15. WILKINSON J. H. : *Rounding errors in algebraic processes*, HMSO, London, 1963.