A Study of Iteration Formulas for Root Finding, Where Mathematics, Computer Algebra and Software Engineering Meet

Gaston H. Gonnet

Abstract. For various reasons, including speed code simplicity and symbolic approximation, it is still very interesting to analyze simple iteration formulas for root finding. The classical analysis of iteration formulas concentrates on their convergence near a root. We find experimentally, that this information is almost useless. The (apparently) random walk followed by iteration formulas before reaching convergence is the dominating factor in their performance. We study a set of 29 iteration formulas from a theoretical and a practical point of view. We define a new property of the formulas, their far-convergence, in an effort to explain their behaviours. Extensive experimentation finding polynomial roots, shows that there are extreme differences in performance of seemingly similar iterators. This is a surprising result. We use this experimental approach to select the most effective performer, which is Laguerre's method. The best companion (second method) to handle the failures of Laguerre's is a new method which is an adaptation of Halley's method to multipoint computation. The little-known Ostrowski's method comes out with one of the best performances. We also find that an unknown simple variant of Newton's method, behaves much better than Newton's method itself, which behaves very poorly. This shows that sometimes it pays to modify a method to improve its far-convergence. Various performance curiosities cannot be explained in terms of neither order of convergence and are probably caused by the paths that the methods force on the iteration values. The study of these random paths is an open problem, probably beyond our present tools.

1. Introduction

We consider the problem of finding a numerical approximation to a root. For simplicity and uniformity reasons we will approximate roots of polynomials. We use $p(x)$ to denote the univariate polynomial of degree d. A root α, $p(\alpha) = 0$, may be real or complex. To avoid trivial cases, we will assume that $d \geq 3$ and $p(0) \neq 0$. A single point iteration formula, can be written as

$$x_{n+1} = x_n - \phi(x_n)$$

where $\phi(\alpha) = 0$. The error of the iteration at step n is

$$\varepsilon_n = x_n - \alpha\,.$$

We have noticed that all single-point iteration formulas that we studied can be expressed in terms of two values:

$$\delta = p(x)/p'(x)$$

and

$$\eta = \frac{p(x)p''(x)}{p'(x)^2}\,.$$

For example, Newton's method has $\phi = \delta$ and Halley's method has $\phi = \delta/(1-\eta/2)$. Both $\delta \to 0$ and $\eta \to 0$ when we are converging to a simple root.

1.1. Near- and far-convergence

The standard definition of order of convergence [4, 7, 6] is based on the behaviour of the iteration in the neighbourhood of a root. That is

$$\varepsilon_{n+1} = x_{n+1} - \alpha = O((x_n - \alpha)^\gamma) = O(\varepsilon_n^\gamma)$$

where α is the searched root, x_n is the n^{th} approximation to the root and ε_n is the error on the n^{th} iteration. A method which verifies the above for maximal γ is said to have order of convergence γ. We will call this order of convergence, *near-convergence*, as it refers to the behaviour at the limit when $x_n \to \alpha$, or when x_n is near α.

We define *far-convergence*, as the convergence order, the same as above, but in the limit when $|x_n - \alpha| \to \infty$. For example, the Newton iteration for the approximate computation of $\sqrt{N}$

$$x_{n+1} = \frac{N/x_n + x_n}{2}$$

has near-convergence of order 2 (quadratic convergence)

$$x_{n+1} - \sqrt{N} = \frac{(x_n - \sqrt{N})^2}{2\sqrt{N}} + O((x_n - \sqrt{N})^3)$$

and far-convergence of order 1 (linear convergence with constant factor 1/2) since

$$\varepsilon_{n+1} = \varepsilon_n/2 + O(1)$$

when $|x_n - \sqrt{N}| = |\varepsilon_n| \to \infty$. The Newton iteration for square roots can be considered very reliable as it converges to a root from any starting point due to its properties of far- and near-convergence (although not at the same speeds).

In this paper we are studying iteration formulas to be used for approximating complex roots of polynomials. In this context, the following can be asserted.

- Any number of derivatives of the function can be computed, each one costing about the same to compute.
- We are interested in working with multiple precision in the context of a computer algebra system, and hence the cost of evaluating $p(x)$ and its derivatives dominates the cost of each iteration.

- When convergence is achieved, an order $\gamma = 1.6$ or $\gamma = 2$ or $\gamma = 3$ makes little difference in the total number of iterations required. However, when the method fails to converge and the values wander around the complex plane, tens or hundreds of iterations can be wasted. From this we conclude that far-convergence is more important than near-convergence.
- Our view is pragmatic. We are searching for an iteration which delivers the best overall efficiency over a sufficiently large sample of polynomials.

2. Iteration Generators

It is possible to generate an infinite number of iteration formulas with a given order of near-convergence. These iterations can be single-point or multi-point. Methods for generating iteration formulas are quite straightforward, we will describe one for single-point direct methods.

Let $A(a_1, a_2, \ldots, a_k, x)$ be an arbitrary function of x with parameters a_1, $a_2, \ldots, a_k$. E.g.

$$A(a_1, a_2, a_3, x) = a_1 + \frac{a_2}{x - a_3}\,.$$

For most functions A with k parameters, it is possible to generate an iteration formula with order of convergence k. This is done through the following steps. Let x_n be an approximation of a zero of $p(x)$. At this point we compute k equations

$$\begin{aligned} A(a_1, a_2, \ldots, a_k, x_n) &= p(x_n) \\ A'_x(a_1, a_2, \ldots, a_k, x_n) &= p'_x(x_n) \\ &\vdots \\ A^{(k-1)}_x(a_1, a_2, \ldots, a_k, x_n) &= p^{(k-1)}_x(x_n)\,. \end{aligned}$$

These equations are viewed as equations in the parameters $a_1, a_2, \ldots, a_k$. We solve these equations and obtain the parameters in terms of x_n, i.e. $a_1(x_n)$, $a_2(x_n)$, etc. Plugging these parameters in A we obtain a function $A(a_1(x_n)$, $a_2(x_n), \ldots,\ a_k(x_n), x)$ which is a k-order approximation to $p(x)$ at x_n. Let x_{n+1} be the zero of $A(\ldots, x)$, i.e. $A(a_1(x_n), a_2(x_n), \ldots, a_k(x_n), x_{n+1}) = 0$, then x_{n+1} becomes our next approximation to the root. The iteration formula is defined by the computation producing x_{n+1} from x_n. Normally we can find a closed form expression of x_{n+1} in terms of $p(x)$ and x_n. To achieve this, the function A is chosen so that the solution of the system of equations can be expressed in closed form and that the zero of A is easy to compute, i.e. $A(\ldots, x) = 0$ is easy to solve. For the above example,

$$\begin{aligned} A(a_1, a_2, a_3, x_n) &= a_1 + \frac{a_2}{x - a_3} = p(x_n)\,, \\ A'_x(a_1, a_2, a_3, x_n) &= -\frac{a_2}{(x_n - a_3)^2} = p'_x(x_n) \end{aligned}$$

and

$$A''_x(a_1, a_2, a_3, x_n) = \frac{2a_2}{(x_n - a_3)^3} = p''_x(x_n)$$

from which we derive

$$a_2 = -\frac{4p'(x_n)^3}{p''(x_n)^2}$$
$$a_3 = \frac{2p'(x_n)}{p''(x_n)} + x_n$$

and

$$a_1 = p(x_n) - \frac{a_2}{x_n - a_3}$$
$$A(a_1, a_2, a_3, x_{n+1}) = a_1 + \frac{a_2}{x_{n+1} - a_3} = 0 \implies x_{n+1} = a_3 - \frac{a_2}{a_1}$$

Simplifying and rearranging we obtain

$$x_{n+1} = x_n - \frac{p(x_n)}{p'(x_n)} \frac{1}{(1 - \frac{p(x_n)p''(x_n)}{2p'(x_n)^2})} = x_n - \frac{\delta}{1 - \eta/2}$$

which is Halley's method [1]. We can say that approximating $p(x)$ by $a_1 + \frac{a_2}{x - a_3}$ generates Halley's iteration.

Table 1 shows the iteration formulas tested and the approximation formulas which generate them. Iterations 2 to 6 are all special cases of $\delta(1 + \frac{\eta}{2a})^a$. This iteration gives third order near-convergence for any value of a. In all cases, it corresponds to an inverse interpolation approximation

$$x \approx (a_1 + a_2 p(x))(1 + a_3 p(x))^a$$

Halley's, Ostrowski's [5] and inverse quadratic interpolation are special cases of this formula. Methods 10 to 17 are derived from Laguerre's method by assuming that the approximation is of a fixed degree (2 to 9). For degree 2, this coincides with Euler's method or direct quadratic interpolation. Many variations of Laguerre's method were tested because all of them have a very good behaviour, in particular the optimal number of successes is achieved by iterator 15. Methods 18 to 25 were artificially created. 18 and 19 are truncated taylor series in η of method 10. Methods 20 to 25 are perturbations of other methods in their $O(\eta^2)$ term (in $O(\delta^2)$ for 25) so that the order of far-convergence becomes $O(1)$ instead of $O(\eta)$. Methods 26 to 29 are multipoint methods based on similar approximations. The Hansen-Patrick methods [2], generated by

$$\frac{\delta(a+1)}{a + \sqrt{1 - (a+1)\eta}}$$

for fixed a, are not included since they can be generated from Laguerre's formula, by setting $a = 1/(d-1)$.

Table 2 shows the near- an far-convergence for each of the methods. For both last columns, the expression represents the asymptotic value of ε_{n+1} in terms of ε_n. Only the most significant term is shown. For the near convergence, the exponent of ε_n is the classical order of convergence. $p' = p'_x(\alpha)$, $p'' = p''_x(\alpha)$ and

	Iteration formula	Generating Approximation	Notes
1	δ	$p(x) \approx a_2 (x - a_1)$	Newton
2	$\frac{\delta}{1-1/2\eta}$	$p(x) \approx \frac{x-a_1}{a_2+a_3 x}$	Halley
3	$\frac{\delta}{\sqrt{1-\eta}}$	$x \approx \frac{a_1+a_2 p(x)}{\sqrt{1+a_3 p(x)}}$	Ostrowski
4	$\frac{\delta}{\sqrt[3]{1-3/2\eta}}$	$x \approx \frac{a_1+a_2 p(x)}{\sqrt[3]{1+a_3 p(x)}}$	
5	$\delta\sqrt{1+\eta}$	$x \approx (a_1 + a_2 p(x)) \sqrt{1 + a_3 p(x)}$	
6	$\delta (1 + 1/2\eta)$	$x \approx a_1 + a_2 p(x) + a_3 p(x)^2$	Inverse quadratic interpolation
7	$\frac{\delta(1-\delta+1/2\eta)}{1-\delta+\delta^2-1/2\eta\delta}$	$x \approx 1 - \frac{1}{(a_1+a_2 p(x)+a_3 p(x)^2)}$	
8	$\frac{\delta\left(2+\sqrt{1+\eta}\right)}{3-\eta}$	$p(x)^2 \approx \frac{(x-a_1)^2}{a_2+a_3 x}$	
9	$\frac{\delta d}{1+\sqrt{(d-1)^2-d(d-1)\eta}}$	$p(x) \approx (x - a_1) (a_2 x + a_3)^{d-1}$	Laguerre
10	$\frac{2\delta}{1+\sqrt{1-2\eta}}$	$p(x) \approx (x - a_1) (a_2 x + a_3)$	Euler, direct quadratic interpolation
11-17	$\frac{\delta k}{1+\sqrt{(k-1)^2-k(k-1)\eta}}$	$p(x) \approx (x - a_1) (a_2 x + a_3)^k$	Laguerre with fix degree $k = 3..9$
18	$\delta (1 + \eta/2 + \eta^2/2)$	artificial	matching 10 to order 2 in η
19	$\frac{\delta}{1-\eta/2-\eta^2/4}$	artificial	inverse matching inverse of 10 to order 2 in η
20	$\frac{\delta}{1-\eta/2-\eta^2/8}$	artificial	inverse matching inverse of 3 to order 2 in η
21	$\delta / \left(1 - \eta/2 - \frac{d\eta^2}{2(d-1)}\right)$	artificial	using degree to force 2 to have far convergence
22	$\frac{\delta}{\sqrt{1-\eta-\frac{\eta^2}{d-1}}}$	artificial	using degree to force 3 to have far convergence
23	$\delta\sqrt{1+\eta+\frac{d\left(d^2+d-1\right)\eta^2}{d-1}}$	artificial	using degree to force 5 to have far convergence
24	$\delta\left(1+\frac{\eta}{2}+\frac{(2d-1)d\eta^2}{2(d-1)}\right)$	artificial	using degree to force 6 to have far convergence
25	$\delta / \left(1 - \frac{(d-1)\delta}{x}\right)$	artificial	Perturbing Newton with degree for good far convergence
26	$\frac{p_n \Delta x_{01}}{\Delta p_{01}}$	$p(x) \approx a_2(x - a_1)$	Secant
27	$\frac{p_n}{\Delta p_{12}} \left(\frac{p_{n-1}\Delta x_{02}}{\Delta p_{02}} - \frac{p_{n-2}\Delta x_{01}}{\Delta p_{01}}\right)$	$x \approx a_1 + a_2 p(x) + a_3 p(x)^2$	Multipoint inverse quadratic interpolation
28	$\frac{2b_0}{b_1(1+\sqrt{1+4b_2 b_0/b_1^2})}$ $b_0 = p_n \Delta x_{02} \Delta x_{12} \Delta x_{01}$ $b_1 = \Delta p_{01} \Delta x_{02}^2 - \Delta p_{02} \Delta x_{01}^2$ $b_2 = p_{n-1} \Delta x_{02} - p_n \Delta x_{12} - p_{n-2} \Delta x_{01}$	$p(x) \approx (x - a_1) (a_2 x + a_3)$	Muller
29	$\frac{p_n \Delta p_{02} \Delta x_{02} \Delta x_{01}}{p_n \Delta p_{12} x - p_{n-1} \Delta p_{02} x_{n-1} + p_{n-2} \Delta p_{01} x_{n-2}}$	$p(x) \approx \frac{x-a_1}{a_2+a_3 x}$	Multipoint Halley

TABLE 1. Iteration formulas.

	near-convergence		far-convergence	
	order	ε_{n+1}	order	ε_{n+1}
1	2	$\frac{p''}{2p'}\varepsilon_n^2$	1	$\frac{d-1}{d}\varepsilon_n$
2-6	3	$\frac{3(3a+1)p''^2-4ap'p'''}{24ap'^2}\varepsilon_n^3$	1	$\left(1-\frac{\left(1+\frac{d-1}{2da}\right)^a}{d}\right)\varepsilon_n$
7	3	$\frac{6p'^2-6p'p''+3p''^2-p'p'''}{6p'^2}\varepsilon_n^3$	1	ε_n
8	3	$\frac{9p''^2-4p'p'''}{24p'^2}\varepsilon_n^3$	1	$\frac{2d-1-\sqrt{2-1/d}}{2d+1}\varepsilon_n$
9	3	$\frac{3(d-2)p''^2-4(d-1)p'p'''}{24(d-1)p'^2}\varepsilon_n^3$	0	$-\alpha+\frac{-c_{d-1}}{dc_d}$ $+\frac{\sqrt{(d-1)\left((d-1)c_{d-1}^2-2dc_dc_{d-2}\right)}}{dc_d}$
10-17	3	$\frac{3(k-2)p''^2-4(k-1)p'p'''}{24(k-1)p'^2}\varepsilon_n^3$	1	$\left(1-\frac{k}{d+\sqrt{-d(d-k)(k-1)}}\right)\varepsilon_n$
18	3	$-\frac{p'''}{6p'}\varepsilon_n^3$	1	$\frac{(d-1)\left(2d^2-2d+1\right)}{2d^3}\varepsilon_n$
19	3	$-\frac{p'''}{6p'}\varepsilon_n^3$	1	$\frac{d^2-1}{d^2+4d-1}\varepsilon_n$
20	3	$\frac{3p''^2-4p'p'''}{24p'^2}\varepsilon_n^3$	1	$\frac{3d^2-2d-1}{3d^2+6d-1}\varepsilon_n$
21	3	$\frac{-3(d+1)p''^2-2(d-1)p'''p'}{12p'^2(d-1)}\varepsilon_n^3$	0	$-\alpha-\frac{c_{d-1}}{c_d}$
22	3	$\frac{3(d-5)p''^2-4(d-1)p'''p'}{24p'^2(d-1)}\varepsilon_n^3$	0	$-\alpha-\frac{c_{d-1}}{c_d}$
23	3	$\frac{-(12d^3+12d^2-27d+15)p''^2-4(d-1)p'''p'}{24p'^2(d-1)}\varepsilon_n^3$	0	$-\alpha-\frac{c_{d-1}}{c_d}$
24	3	$\frac{-(d-1)p'p'''-(6d^2-6d+3)p''^2}{6p'^2(d-1)}\varepsilon_n^3$	0	$-\alpha-\frac{c_{d-1}}{c_d}$
25	2	$\frac{\alpha p''-2p'(d-1)}{2p'\alpha}\varepsilon_n^2$	0	$-\alpha-\frac{c_{d-1}}{c_d}$
26	1.618...	$\frac{p''}{2p'}\varepsilon_n\varepsilon_{n-1}$		
27	1.839...	$\frac{3p''^2-p'p'''}{6p'^2}\varepsilon_n\varepsilon_{n-1}\varepsilon_{n-2}$		
28	1.839...	$-\frac{p'''}{6p'}\varepsilon_n\varepsilon_{n-1}\varepsilon_{n-2}$		
29	1.839...	$-\frac{3p''^2-2p'p'''}{12p'^2}\varepsilon_n\varepsilon_{n-1}\varepsilon_{n-2}$		

TABLE 2. near- and far -convergence of iteration formulas.

$p''' = p'''_x(\alpha)$ denote the derivatives of $p(x)$ at the root. $\Delta x_{ij} = x_{n-i} - x_{n-j}$, $\Delta p_{ij} = p(x_{n-i}) - p(x_{n-j})$, p_n stands for $p(x_n)$ and c_i is the coefficient of x^i in $p(x)$. For both columns, a smaller coefficient for ε_n is better. For the first column, a higher exponent of ε_n is better, as $\varepsilon_n \to 0$, while for the last column a lower exponent is better, as $\varepsilon_n \to \infty$. Far-convergence cannot be easily defined for multistep methods and hence was not computed.

3. The Search

Although the method described here could be used for any type of problems, or for any order of convergence, we concentrate on methods with cubic convergence ($\gamma = 3$) used on polynomials. As we said earlier, we will take a very pragmatic approach and try as many formulas as possible against a sufficiently large number of problems and select the best performing one(s).

Since the target application of this method is in Maple, we will test the algorithms using the same system so that the complexity measures (mainly time) are compatible.

3.1. The problem set

The problem set should be large and open-ended, so that we avoid selecting methods which happen to be lucky with a particular sample set. This is relevant if we are going to test many iteration formulas. We tested each method with 100000 polynomials so that critiques with respect to sample size may be silenced. The best performing formulas (3,9-17,22 and 25) were tested against 200000 polynomials. The sample is generated at random, so that it is open-ended, i.e. we could test these methods for 10 times more problems if desirable. In all cases the polynomials have integer coefficients. We generate the polynomials in 4 different groups. In all cases, a random variable distributed as $U(a, b)$ denotes a uniform random distribution between the values a and b inclusive. All polynomials are generated with a random degree distributed as $U(3, 20)$.

(a) 40% of the polynomials with random $U(-10^{10}, 10^{10})$ coefficients. The number of non-zero coefficients is also random $U(3, deg(p) + 1)$. This is an example of such a polynomial.

$$422434762x^7 - 8643406254x^6 + 6868525159x^5 + 9284921461x^3 + 3505658871$$

(b) 30% of the polynomials with coefficients which are random integers $U(-10^k, 10^k)$ and k is $U(0, 40)$ for each coefficient. The number of non-zero coefficients is $U(2, deg(p) + 1)$. E.g.

$$\begin{aligned}&- 708032792x^{19} + 7736163542925258837 10x^{17}\\&+ 9069436236132014192 7817x^{15} + 287243520412466889x^{13} + 990598934x^{12}\\&+ 642099534061826306764360057825 1x^{10} - 54255029639x^9\\&- 539443581303941066250263794671941160274\end{aligned}$$

(c) 10% of the polynomials as in (b) above, but with random gaussian integers (complex integers) as coefficients. This is done by multiplying each coefficient by i^k where k is a random integer $U(0, 1)$. E.g.

$$\begin{aligned}&9720195260033566442710141098991462068038ix^8\\&\qquad - 6779x^3 + 33787241729985 + 606i\end{aligned}$$

(d) 20% of the polynomials which are a minor perturbation of a product of powers of simpler polynomials. Although the degree of the polynomial is $U(3, 20)$, each factor of these polynomials has random $U(1, 3)$ degree and is powered to a random exponent so that it does not exceed the expected degree of the final polynomial. The perturbation is obtained by adding $\pm x^k$ where k is $U(0, deg(p))$. E.g.

$$(9887x + 446)^6 (-273x + 12)^4 \left(-3230x^3 - 20\right)^2 (3178x + 6) - x^{13}$$

The polynomials in group (a) are truly random polynomials but are fairly easy to resolve by all methods and present a lesser challenge. Thus we have weighted them only 40% in the sample. The groups (b) and (c), represent cases where the large differences in the magnitude of the coefficients could lead to extreme placements of the roots. Roots could lie very close together, or have very small imaginary or real parts, or have very different magnitudes. In our experience these polynomials reveal the weaknesses of the iteration formulas, and consequently we have weighted these two groups as much as group (a). Notice that (c) should not behave too differently from (b), it was included just to insure that polynomials with complex coefficients do not cause additional problems. The last group, (d), contains polynomials which will have roots very close together, i.e. very small perturbations of non square-free polynomials. This is not so uncommon in practice, and again, most iteration formulas have difficulties in this situation.

4. Test for each Polynomial

For each polynomial, and for each formula, the iteration is started at the same point.

$$x_0 = (1+i)\sqrt{2}\max_k \sqrt[k]{\left|\frac{c_{d-k}}{c_d}\right|}$$

where d is the degree of the polynomial and c_i is the coefficient of x^i in $p(x)$. This is a point at 45 degrees on a complex circle which encloses all roots of the polynomial.

All computations were done with increasing precision, the initial precision is identical to the number of digits in the largest coefficient and it is increased by one decimal digit after each iteration. This reflects the accepted practice in variable precision computer algebra systems. One of the reasons for which iterations fail is the ill-conditioning of the polynomial which may require additional precision to resolve.

Convergence to a root is decided from the computed values and estimates of the errors of the last step. For multipoint methods with error formulas $\varepsilon_{n+1} \approx \alpha\varepsilon_n\varepsilon_{n-1}$ or $\varepsilon_{n+1} \approx \alpha\varepsilon_n\varepsilon_{n-1}\varepsilon_{n-2}$, in the neighbourhood of a root and with convergence, the estimate for ε_{n+1} is

$$\varepsilon_{n+1} \approx \frac{\varepsilon_n^2}{\varepsilon_{n-2}} \text{ or } \varepsilon_{n+1} \approx \frac{\varepsilon_n^2}{\varepsilon_{n-3}}.$$

The convergence criteria, once x_{n+1} has been computed, is to test whether the relative error of x_{n+1} is small enough, or

$$\left|\frac{\varepsilon_{n+1}}{x_{n+1}}\right| < M_\varepsilon$$

which gives the complete condition

$$|\varepsilon_n| < |\varepsilon_{n-1}| < |\varepsilon_{n-2}| < |\varepsilon_{n-3}| \text{ and } \left|\frac{\varepsilon_n^2}{\varepsilon_{n-2}x_{n+1}}\right| < M_\varepsilon$$

where M_ε is the machine epsilon, or the relative precision at which we want to compute the roots. We do not know the exact ε_{n-i}, but we can approximate them with $\varepsilon_{n-i} \approx x_{n-i} - x_{n+1}$. In the neighbourhood of a root and with superlinear convergence, this approximation is sufficient.

For single-point methods, with error term of the form $\varepsilon_{n+1} \approx \alpha\varepsilon_n^\gamma$ we compute $\varepsilon_{n+1} \approx \varepsilon_n^{\gamma+1}/\varepsilon_{n-1}^\gamma$ and obtain the first condition. For these iterations we already computed the derivative of the polynomial at each step, so we can use these derivatives to estimate the error at x_{n+1}

$$p'(x_n)(x_n - \alpha) \approx p(x_n) - p(\alpha) = p(x_n)$$

$$\varepsilon_n = p(x_n)/p'(x_n) = \delta_n\,.$$

We can use both estimates ($\varepsilon_{n-i} = \delta_{n-i}$ and $\varepsilon_{n-i} = x_{n-i} - x_{n+1}$) to derive a safer criteria for convergence

$$|\varepsilon_n| < |\varepsilon_{n-1}| \text{ and } \left|\frac{\varepsilon_n^{\gamma+1}}{\varepsilon_{n-1}^\gamma x_{n+1}}\right| < M_\varepsilon \text{ and } \left|\frac{\delta_n^{\gamma+1}}{\delta_{n-1}^\gamma x_{n+1}}\right| < M_\varepsilon\,.$$

No acceleration of the iterations is attempted (as it is common practice when the method shows linear convergence), as this is one of the features which we want to measure. We want to evaluate the iterators on their exclusive own merits.

We compute only one root for each polynomial, as it is cheaper and safer to compute a new polynomial than to deflate one.

The best iteration schemes use an average of about 10 iterations per root. We let each method run at most 50 iterations. If the algorithm runs for 50 iterations and does not converge, we consider it has failed. Failures to compute the iteration formula (like divisions by zero, very large floating exponents and repeated values) are also counted as failures. In the case of failures, the number of iterations and time consumed is accounted up to the point of failure. Each failure due to lack of convergence is very costly in time, and hence total time becomes an excellent compound measure for the iteration formulas.

We measure, for each polynomial and each method, the number of iterations, the time spent and the failures. Notice that the relation between number of iterations and total time spent is not necessarily linear due to the increasing precision of the computation. Time and number of failures are the most important measures.

5. Conclusions

The whole computations in this paper used about 11,984 hours of CPU time on a small subnetwork of HP and Sun workstations. The results are shown in table 3 from which we can reach several conclusions.

	iterations					time					failures (%)				
	a	b	c	d	all	a	b	c	d	all	a	b	c	d	all
1	18.7	33.5	32.3	40.4	28.8	1.00	3.19	3.13	4.14	2.50	5.10	48.21	45.35	57.54	32.55
2	10.1	26.7	25.6	31.9	21.0	.66	3.90	3.82	5.24	2.86	1.87	34.03	32.33	28.65	19.92
3	*5.7*	19.1	18.4	22.9	14.4	*.36*	3.04	2.97	3.94	2.14	*.00*	19.48	18.99	5.95	8.93
4	15.9	19.3	19.2	35.5	21.2	2.56	5.24	5.23	9.96	5.11	1.36	8.68	8.38	51.05	14.20
5	15.2	30.6	29.4	36.4	25.5	1.27	4.81	4.70	6.19	3.66	4.58	42.02	40.00	43.82	27.20
6	15.0	30.9	29.3	35.9	25.4	1.13	4.40	4.25	5.64	3.32	5.49	42.76	40.12	41.71	27.38
7	13.9	38.8	38.7	42.7	29.6	1.10	6.16	6.24	6.77	4.27	7.08	68.60	68.90	71.41	44.59
8	11.5	28.0	27.0	33.8	22.5	.89	4.66	4.54	6.11	3.43	1.86	36.76	35.08	34.76	22.23
9	6.2	13.6	13.7	*15.0*	*10.9*	.49	2.20	2.30	*2.59*	1.60	4.30	13.60	13.00	5.18	8.14
10	11.4	15.0	15.7	29.0	16.4	.94	2.19	2.34	5.25	2.32	.33	7.82	9.51	29.62	9.35
11	10.3	14.1	14.5	27.4	15.3	.84	2.05	2.17	5.06	2.18	.28	7.49	8.70	25.91	8.41
12	9.6	13.4	13.2	25.9	14.3	.78	1.95	1.92	4.83	2.05	.25	7.20	6.66	22.44	7.41
13	9.0	*12.8*	12.7	24.8	13.7	.72	1.86	1.84	4.62	1.95	.21	*7.01*	*6.54*	20.26	6.89
14	8.4	13.0	12.7	23.8	13.3	.67	1.88	1.83	4.43	1.90	.18	7.78	7.06	17.68	6.65
15	7.9	13.0	12.6	22.7	12.9	.62	1.86	1.81	4.23	1.83	.16	7.84	7.25	15.25	*6.19*
16	7.5	13.4	13.0	22.1	12.7	.58	1.95	1.90	4.09	1.82	.14	9.20	8.29	13.82	6.41
17	7.1	13.4	13.3	21.4	12.5	.53	1.96	1.95	3.94	1.78	.08	9.59	9.53	12.07	6.28
18	14.3	29.0	27.8	32.9	23.8	1.12	4.27	4.16	5.40	3.22	6.58	38.20	36.10	31.88	24.08
19	9.7	24.0	22.9	26.1	18.6	.67	3.58	3.50	4.38	2.57	6.03	28.25	27.27	14.54	16.52
20	9.4	24.9	24.0	29.3	19.5	.63	3.77	3.70	4.98	2.75	3.63	30.88	29.59	22.05	18.08
21	30.1	31.8	31.7	27.6	30.3	3.06	4.53	4.63	4.94	4.03	60.97	51.71	51.24	38.19	52.66
22	9.1	22.6	22.3	17.3	16.1	.71	3.51	3.58	2.68	2.23	4.75	13.12	13.22	*4.99*	8.16
23	31.9	35.8	36.0	45.5	36.2	2.72	5.62	6.12	7.49	4.88	89.81	88.89	88.09	93.44	90.09
24	30.4	34.7	34.8	42.4	34.5	2.19	4.95	5.36	6.55	4.21	85.91	85.23	84.96	87.04	85.84
25	17.1	21.2	21.6	25.7	20.5	1.08	2.24	2.30	3.29	1.99	10.42	19.94	21.17	24.58	17.18
26	19.2	35.4	33.8	42.7	30.2	.82	2.59	2.50	3.14	1.98	14.90	58.20	54.81	67.08	42.32
27	19.6	13.3	*12.5*	27.3	18.5	1.21	*1.07*	*1.00*	2.82	*1.47*	47.46	63.56	63.01	56.61	55.68
28	13.1	25.9	24.8	39.2	23.3	1.35	5.02	4.81	7.07	3.94	2.56	38.14	35.68	59.68	27.97
29	15.9	17.2	16.6	31.5	19.5	1.04	1.96	1.88	3.56	1.90	17.23	54.98	55.26	45.39	37.99

TABLE 3. Average values for each method (row) and type of polynomial.

The first conclusion is that there is a substantial difference in performances. Not only in the time consumed, but also in the number of failures. The effectiveness of the best method compared to the effectiveness of the worst is in a ratio of approximately 1:30. We define effectiveness as the fraction of solved polynomials divided time. Methods with apparently very similar characteristics perform dissimilarly. Order of near-convergence does not appear to be a criteria to base any consistent judgement. In particular the order of the worst performer is equal to the best performer and is better than the one of the third best performer!

Not surprisingly, the overall winner is Laguerre's iteration. It is a winner in effectiveness and in 3 of the 15 categories (columns of the table). Entries in italics indicate the best value for the given column. It only looses by 2% to the best in number of failures. Other methods of the Laguerre's family are also very effective, in particular for high fixed degree. One of these methods, 15, is the one producing the fewest failures.

The relatively unknown Ostrowski's method, which comes next in efficiency, is remarkable in one way. It does not fail with *any* of the polynomials of group a. We continued to run Ostrowski's method against polynomials of type a, and we

run it with over one million polynomials without failure. There is something quite remarkable about its convergence which escapes our analysis, and which places it in a unique category with respect to all other methods.

Method 25 ranks next in effectiveness. This is a big surprise. Method 25 is a modification of Newton's method, which improves the far-convergence order and outperforms Newton's in 13 of the 15 categories. The number of failures is cut in half. This shows that modifying a method to improve its far-convergence, while keeping its near-convergence, could improve the method significantly.

The artificially generated method 22 (a modification of Ostrowski's method to improve far-convergence) comes next in efficiency, still outperforming all other more famous methods. It produces a very small number of failures, and it appears to be the best suited for problems in the group d.

Again, for reasons unknown to us, methods 23 and 24, which were supposed to be improvements on methods 5 and 6, behave disastrously. In particular much worse than 5 and 6. Notice that the same type of change was done for 22 (from 3), in which case they remained about the same and for 25 (from 1) which produced a significant improvement.

All these curiosities and oddities indicate that the overall performance of these methods is not tied to the convergence part alone, but rather to the apparently random walks over the complex plane. These walks are the most expensive part of the computation and depend on subtleties of the iterations, at present beyond our understanding. Hence our selection method here is empirical.

The effectiveness of these walks is dramatically different for different methods. Our methods of analysis cannot explain this behaviour which is probably one of the most difficult open problems in the area.

For samples of 100,000 or 200,000 polynomials, the confidence intervals of the values in the tables are very small. For example, the totals for Laguerre's method have 95% confidence intervals $10.94 \pm .12$, $1.604 \pm .027$ and $8.14 \pm .24$ respectively. The difference in efficiency with the second most effective method (17) are well outside these confidence intervals.

5.1. The best companion for Laguerre

Once that we have observed that the best performer is Laguerre's, and that this method still fails on one out of 12 polynomials, the immediate next question is which method will handle the largest portion of problems where Laguerre fails. If there is a pattern in the problems being solve, then there is a pattern (complementary) on the failed ones, and hence a particular method may be well suited to handle them.

Table 4 contains the results of running problems which were not resolved by Laguerre. For Laguerre itself, the runs were done starting at a different point ($x_0 e^{2\pi i \phi}$, where ϕ is the golden ratio). This gives an indication of how dependent the failures were from an initial unlucky value. For this run we are also interested in the effectiveness, that is number of problems solved divided total time.

	iterations					time					failures (%)				
	a	b	c	d	all	a	b	c	d	all	a	b	c	d	all
1	22.7	30.8	32.3	48.6	31.7	.87	2.58	2.85	6.35	2.75	19.11	41.58	45.69	85.06	43.17
2	12.9	23.4	24.7	42.5	23.9	.61	3.03	3.40	9.11	3.37	10.78	33.03	36.70	57.99	32.18
3	*5.4*	18.1	20.7	*32.2*	17.7	*.25*	2.79	3.36	7.86	3.00	*.00*	26.90	32.94	*23.57*	*21.78*
4	17.7	27.0	27.7	49.7	28.1	2.38	7.25	7.58	16.87	7.53	*.00*	26.41	28.62	96.38	30.30
5	18.1	26.9	28.2	46.1	27.8	1.09	3.79	4.16	9.99	4.09	14.22	37.09	40.00	73.16	37.43
6	20.0	28.2	30.1	45.7	29.1	1.12	3.56	4.02	9.24	3.86	19.61	41.24	45.78	71.18	41.32
7	11.8	29.5	31.2	47.8	28.4	.69	4.29	4.80	10.29	4.39	.07	44.72	49.54	86.11	41.47
8	14.0	24.5	25.8	44.1	25.0	.78	3.62	4.02	10.23	3.95	10.42	34.84	37.61	64.18	33.96
9	39.8	45.5	37.6	46.1	43.0	4.43	8.52	7.41	10.24	7.70	75.22	93.84	76.97	83.20	85.73
10	12.3	22.6	24.9	46.1	23.8	.74	3.43	3.99	10.32	3.85	*.00*	25.64	31.93	70.71	27.13
11	11.5	22.2	24.1	46.0	23.4	.69	3.36	3.85	10.61	3.82	*.00*	25.76	30.83	73.51	27.37
12	10.9	21.7	22.6	44.5	22.5	.64	3.29	3.52	10.46	3.70	*.00*	*25.45*	27.43	70.95	26.32
13	10.5	21.4	22.2	44.8	22.3	.61	3.24	3.46	10.45	3.66	*.00*	25.92	27.52	74.45	27.02
14	10.1	21.2	21.8	43.8	21.9	.58	3.20	3.39	10.28	3.60	*.00*	25.73	*27.16*	73.86	26.79
15	9.6	20.8	21.6	41.6	21.3	.55	3.15	3.37	10.10	3.54	*.00*	25.73	27.98	68.61	26.24
16	9.0	20.4	21.2	39.6	20.6	.51	3.10	3.30	9.87	3.47	.14	25.92	27.71	63.71	25.68
17	8.6	20.0	21.1	38.4	20.1	.49	3.04	3.32	9.72	3.42	.22	25.76	28.17	61.14	25.36
18	16.9	26.0	27.6	43.2	26.6	.95	3.43	3.85	9.27	3.74	16.95	38.97	42.48	62.19	37.92
19	14.6	24.5	26.2	36.2	24.2	.86	3.39	3.82	8.44	3.58	19.54	39.74	43.85	36.76	35.76
20	11.9	22.6	24.1	40.0	22.9	.59	3.10	3.50	9.09	3.41	11.35	34.04	37.98	49.01	31.85
21	6.8	33.8	34.4	46.8	29.9	.36	5.21	5.65	7.89	4.61	100.0	84.09	81.38	89.26	87.67
22	45.3	45.6	45.7	44.2	45.4	5.00	7.47	7.90	6.55	6.90	76.29	67.07	67.16	72.58	69.75
23	10.3	24.3	26.8	48.2	24.9	.37	3.13	4.56	8.00	3.42	100.0	96.95	97.16	99.77	98.00
24	12.0	24.9	27.1	48.3	25.6	.41	2.93	4.09	7.59	3.20	99.93	95.81	96.06	99.42	97.19
25	47.8	36.5	35.6	42.0	39.4	3.73	4.33	4.29	7.07	4.55	83.69	63.16	63.58	64.06	67.68
26	19.7	29.0	31.0	49.7	30.1	.72	1.96	2.18	*4.62*	2.08	36.42	51.06	55.14	95.22	54.39
27	17.2	15.8	13.6	43.7	19.4	.90	1.26	1.08	5.90	1.76	37.00	47.98	50.92	82.26	50.61
28	7.8	20.3	22.7	49.6	21.9	.60	3.71	4.28	10.99	4.09	.86	25.82	31.93	95.57	30.63
29	14.9	*13.4*	*11.7*	42.3	*17.2*	.82	*1.16*	*1.01*	6.13	*1.71*	12.72	34.44	37.89	70.83	35.15

TABLE 4. Average values for each method (row) and type of polynomial over the failures of method 9.

The best companion for Laguerre is method 29, a multipoint version of Halley's iteration, which fails on 35% of the residual polynomials. Notice that the statistics for method 29 as a second method are *better*, in all three categories, than as a first method. This means that it is very complementary to Laguerre's method, the polynomials which were harder for Laguerre, are easier for 29.

Using Laguerre a second time, with a different starting value, does not appear to be a good idea at all. It fails 86% of the time, i.e. a different starting value is successful about $1/7^{\text{th}}$ of the time. This somehow *proves* that the failure set has a pattern.

Ostrowski's method (3) is the most successful companion, but it takes more time than method 29, and hence its effectiveness is lower.

5.2. Best companion of Laguerre and 29

Following the same idea, we ran all the methods against the residuals of Laguerre and method 29. The best method for handling these residuals is method 25, the modified Newton method. We omit the table showing these results. Method 25

solves 56% of the remaining polynomials, in a very competitive time. It is by far the most effective method for these residual polynomials.

We ran this process 6 times, to find the most effective sequence of methods to handle these polynomials. When a method is used more than once with the same polynomial, we start the iteration at a different value. The starting value for the j^{th} run of a method is set to $x_0 e^{2\pi i(j-1)\phi}$. Table 5 summarizes the results of these simulations. Each method was selected to have highest effectiveness, that is number of problems solved divided total time. It is obvious from the first three methods already, that each one of them *specializes* for some type of polynomials and that their combined use is much more effective than their repeated use.

prev methods	meth	iter	time	failures (%)	solved/ time	cumulative fail (%)	sample size
	9	10.901±.086	1.597±.019	7.99±.17	57.61	7.99±.17	92450
9	29	17.16±.40	1.710±.060	35.2±1.2	37.92	2.824±.093	82000
9,29	25	24.70±.55	3.398±.098	44.3±1.1	16.39	1.298±.034	246000
9,29,25	3	21.87±.42	5.22±.16	15.32±.89	16.22	.196±.011	492000
9,29,25,3	22	35.85±.43	11.24±.19	20.7±1.4	7.05	.0403±.0027	1640000
9,29,25,3,22	9	36.0±1.1	14.22±.62	52.9±3.1	3.31	.0210±.0012	2460000

TABLE 5. Performance of each method used on the residuals previous methods.

5.3. Conclusions

From this experimental/theoretical study we can derive the following conclusions.

The effectiveness of different iteration formulas cannot be explained by orders or convergence or any other form of analysis presently available.

Different iterations have very different performances, up to a factor of 30 in effectiveness. This cannot be ignored. It appears that each method has its own *personality*.

Newton's iteration formula is one of the poorest performers. An interesting observation on the most popular iteration formula. Laguerre's, Ostrowski's and multipoint-Halley's are the best iterations and should be given special consideration.

Pairs of companion methods are much more effective than one method applied repeatedly.

References

[1] W. Gander. On halley's iteration method. *Americam Mathematical Monthly*. 92(2):131–134, Feb. 1985.

[2] E. Hansen and M. Patrick. *A family of root finding methods*. Numer. Math., 27:257–269, 1977.

[3] P. Henrici. *Elements of Numerical Analysis*. John Wiley, New York, 1964.

[4] P. Henrici. *Essentials of Numerical Analysis*. John Wiley, New York, 1982.

[5] A. M. Ostrowski. *Solution of Equations and Systems of Equations*. Academic Press, New York, 1973.

[6] A. Ralston. *A First Course in Numerical Analysis*. McGraw-Hill, 1965.

[7] J. F. Traub. *Iterative Methods for Solution of Equations*. Prentice-Hall, 1964.

Institut für Wissenchaftlichtes Rechnen,
Eidgenössische TH Zurich-Zentrum,
8092 Zurich, Switzerland
E-mail address: gonnet@inf.ethz.ch