Algorithms Seminar 2001–2002,
F. Chyzak (ed.), INRIA, (2003), pp. 49–54.

Available online at the URL
http://algo.inria.fr/seminars/.

Numerical Elimination, Newton Method and Multiple Roots

Jean-Claude Yakoubsohn
MIP, Université Paul Sabatier, Toulouse (France)

November 19, 2001

Summary by Bruno Salvy

Abstract

Newton's iteration has quadratic convergence for *simple* roots. We present a Newton-based iteration scheme with quadratic convergence for *multiple* roots of systems of analytic functions. This is a report on work in progress.

1. Newton Iteration, Approximate Roots and γ-Theorems

1.1. **Newton Iteration.** Let $f : \mathbb{C}^n \to \mathbb{C}^n$ be an analytic function. Newton's method for solving $f = 0$ consists in approximating f by its linearization at a given point z, whence the equation

$$f(z) + f'(z)(y - z) = 0 \tag{1}$$

from where solving for y yields the following iteration

$$z_{k+1} = N_f(z_k) := z_k - f'(z_k)^{-1} f(z_k). \tag{2}$$

For this method to converge to a root ζ, it is necessary that $f'(\zeta)$ be invertible. The exact domain from where the iteration converges to a solution can have a very complicated fractal structure. However, convergence is usually very fast provided the initial point z_0 be chosen sufficiently close to ζ. This is made more precise in the following [1, Ch. 8].

Theorem 1 (Smale)**.** *Let f be analytic and*

$$\gamma(f, z) := \sup_{k>1} \left(\frac{\left\| f'(z)^{-1} f^{(k)}(z) \right\|}{k!} \right)^{\frac{1}{k-1}}.$$

If $f(\zeta) = 0$ and $f'(\zeta)^{-1}$ exists then for any z such that $\|z - \zeta\| \leq (3 - \sqrt{7})/\big(2\gamma(f, \zeta)\big)$, the sequence defined by $z_0 = z$ and (2) is well defined and satisfies

$$\|z_k - \zeta\| \leq \frac{\|z - \zeta\|}{2^{2^k - 1}}, \qquad k \geq 0. \tag{3}$$

Thus, there is a ball around the root such that starting with any point inside this ball, each step of Newton's iteration decreases the distance to the root quadratically.

An important property of γ in Theorem 1 is that it is actually invariant under unitary changes of coordinates, thus it is related to geometry rather than computation.

Proof. This is the only proof we give in detail. It gives a good idea of the principle of most of the other proofs in this summary.

All the necessary quantities are first expressed in terms of the Taylor coefficients of f at ζ by means of Taylor expansion of f and f' inside (2):

$$N_f(z)-\zeta = f'(z)^{-1}\big(f'(z)(z-\zeta)-f(z)\big) = f'(z)^{-1}f'(\zeta)\sum_{k\geq 1}(k-1)f'(\zeta)^{-1}\frac{f^{(k)}(\zeta)}{k!}(z-\zeta)^k.$$

Set $u=\gamma(f,\zeta)\|z-\zeta\|$, then provided $u<1$, the norm of the sum above is bounded by $\|z-\zeta\|\times\big((1-u)^{-2}-(1-u)^{-1}\big)$. The norm of the remaining product is bounded by considering first its inverse, again via a Taylor expansion: $f'(\zeta)^{-1}f'(z)=1+B$, with $\|B\|\leq(1-u)^{-2}-1$, provided again that $u<1$. Now if $\|B\|$ itself is smaller than 1 (i.e., $u<\big(5-\sqrt{17}\big)/4\approx .2192$), the norm of the inverse can be bounded by the geometric series and this leads to $\big\|N_f(z)-\zeta\big\|\leq\big\|z-\zeta\big\|u/\psi(u)$ where $\psi(u)=1-4u+2u^2$. Thus the Newton iteration converges as soon as $u<\psi(u)$, i.e., $u<\big(5-\sqrt{17}\big)/4$ and the conclusion of the theorem follows from considering $u<\psi(u)/2$. □

A simple corollary of the end of this proof is that two distinct roots ζ and ζ' are at distance at least $\big(5-\sqrt{17}\big)/\big(4\gamma(f,\zeta)\big)$. This last result has been strengthened by Dedieu [2], who obtained $1/2$ instead of .2192 (see below for a proof). Theorem 1 is the basis for the following.

Definition 1. An *approximate root* of $f(z)=0$ is a point such that the sequence defined by $z_0=z$ and (2) is well defined and satisfies (3).

Although the results in Theorem 1 are stated in terms of quantities evaluated at the root, similar manipulations of Taylor expansions lead to variants in terms of quantities evaluated at the current iterate, on which tests can be based [1, Section 8.2].

1.2. **Homotopy.** A remaining problem is to locate approximate solutions.

In the univariate case, this can be achieved by starting from sufficiently many initial points [5].

Another method, which works also in the multivariate case, starts from the map $f_t:[0,1]\times\mathbb{C}^n\to\mathbb{C}^n$ defined by $f_t(x)=f(x)-tf(x_0)$. For $t=1$ a root x_0 is known and then one "follows" the curve from $(1,x_0)$ to $(0,x)$ where x is a root of f. The idea is to partition $[0,1]$ into $t_0=1>t_1>\cdots>t_k=0$ and then apply only one Newton iteration at each t_i: $z_0=x_0$, $z_{i+1}=N_{f_{t_{i+1}}}(z_i)$, for $i=0,\ldots,k-1$. The complexity of this method is related to how small k can be made. This in turn is eventually related to the so-called *fiber distance* of the system along this curve to the discriminant variety (see [1]).

2. Multiple Roots and Clusters in the Univariate Case

Numerically, there is no difference between multiple roots and clusters of roots. Also, the scale of the problem is the only difference between clusters of roots and well-separated roots. Thus, it is important to find an algorithm converging efficiently to clusters of roots. After a cluster has been isolated, either the computation stops and outputs a ball containing the cluster and the number of its elements, or the scaling is refined and the method is applied recursively to converge to each of the roots or subclusters. It turns out that this is possible by exploiting the fact that Newton's iteration does *not* converge *quadratically* to multiple roots. We now review the properties of Newton's method leading to an algorithm [7].

2.1. **Multiplicity and Speed of Convergence.** If ζ is a root of multiplicity m of f, then in the neighbourhood of ζ, one has

$$f(z)\sim f^{(m)}(\zeta)\frac{(z-\zeta)^m}{m!},\qquad f'(z)\sim f^{(m)}(\zeta)\frac{(z-\zeta)^{m-1}}{(m-1)!}.$$

From there it follows that if $m > 1$, the speed of convergence gives information on the value of m:

$$N_f(z) - \zeta = (z - \zeta) - \frac{f(z)}{f'(z)} \sim \left(1 - \frac{1}{m}\right)(z - \zeta). \tag{4}$$

Another consequence of this estimate is that in the neighbourhood of ζ, *the Newton sequence is close to a straight line.*

2.2. **Algorithm.** The idea is to use these properties in three steps: *(i)* compute three iterates x_0, x_1, x_2 and use them to estimate m in view of $(x_2 - x_1)/(x_1 - x_0) \sim 1 - 1/m$; *(ii)* use this value of m to "jump" directly to a better approximation of ζ using (4); *(iii)* control whether this approximation lies in the center of a cluster of m roots using *a posteriori* bounds; if so return the approximation and a radius of a ball containing the cluster, otherwise compute a new iterate x_3 and start again.

2.3. **A Posteriori Bounds.** Recall that Rouché's theorem states that if $\big|f(x) - g(x)\big| < \big|g(x)\big|$ on a circle $|x - z| = r$, then f and g have the same number of zeroes (counted with multiplicity) in the disk with centre z and radius r. Bounds are obtained by considering g defined by $g(x) = \sum_{k \geq m} f^{(k)}(z)(z - x)^k/k!$. If $f^{(m)}(z) \neq 0$, this polynomial has a zero of multiplicity m at z and does not have any other root in a disc of radius at least $R = 1/2\gamma_m(f, z)$, where

$$\gamma_m(f, z) := \sup_{k > m} \left| \frac{m! f^{(k)}(z)}{k! f^{(m)}(z)} \right|^{\frac{1}{k-m}}.$$

This is proved by evaluating f at another zero w using the Taylor expansion of f at z: from $f(w) = 0$ we get

$$\frac{f^{(m)}(z)}{m!}(w - z)^m = \sum_{k > m} \frac{f^{(k)}(z)}{k!}(w - z)^k.$$

Then using the triangular inequality and the definition of γ_m yields the result.

The next step is to find a radius $r \leq R$ such that $|f - g| < |g|$ on the corresponding circle. Rewriting this using the triangular inequality leads to the following.

Theorem 2 (Yakoubsohn)**.** *Let $z \in \mathbb{C}$ and $0 < r < 1/2\gamma_m(f, z)$ be such that*

$$\frac{|f^{(m)}(z)|}{m!} r^m - \sum_{k \neq m} \frac{|f^{(k)}(z)|}{k!} r^k > 0,$$

then f contains m roots counted with multiplicity in the disk of centre z and radius r.

2.4. **Homotopy.** In the same way as in the case of simple roots, it is possible to combine these ideas with a homotopy method to extend the domain of application of this algorithm. A discussion of a way to find an appropriate subdivision $1 = t_0 > \cdots > t_k = 0$ can be found in [7], together with a complexity analysis.

3. Moore–Penrose Inverse

When the number of variables is different from the number of equations, or more generally when $f'(z)$ in (1) is not an isomorphism, Newton's iteration (2) does not apply. It turns out that a simple modification of this iteration gives a process with interesting fixed points both in the underdetermined and overdetermined case. In the underdetermined case, the fixed points are the points of the variety defined by f, while in the overdetermined case, the fixed points are solutions

of the least-square problem associated with the equations. This is achieved by means of the Moore–Penrose inverse. Let A be a linear operator between two Euclidean or Hermitian spaces E and F. The *Moore–Penrose* inverse $A^\dagger$ of A is defined as $A^\dagger = \iota B^{-1}\pi_{\operatorname{Im} a}$, where $\pi_{\operatorname{Im} a}$ is the orthogonal projection on the image $\operatorname{Im} A$ of A, B is the restriction B of A to the orthogonal of its kernel, and ι is the injection of this orthogonal to A. It satisfies $A^\dagger A = \pi_{(\operatorname{Ker} A)^\perp}$, $AA^\dagger = \pi_{\operatorname{Im} A}$.

With this inverse, the Newton iteration can be generalized to the Newton–Gauss iteration:

$$z_{k+1} = N_f(z_k) := z_k - f'(z_k)^\dagger f(z_k). \tag{5}$$

The convergence properties of this iteration are expressed in terms of

$$\gamma^\dagger(f,z) := \sup_{k>1} \left(\frac{\left\| f'(z)^\dagger f^{(k)}(z) \right\|}{k!} \right)^{\frac{1}{k-1}}.$$

The result is parallel to Theorem 1, d denotes the distance.

Theorem 3 (Shub–Smale, Dedieu–Shub)**.** *Let V be the zero-set of f and $\zeta \in V$. If $f'(\zeta)$ is surjective, then for any z such that $|z-\zeta| \le c/\gamma^\dagger(f,\zeta)$, the sequence defined by $z_0 = z$ and (5) is well defined and satisfies $d(z_k,V) \le d(z,V)/2^{2^k-1}$, for $k \ge 0$. Moreover, the sequence converges to a point $Z \in V$ such that $d(z,V) \le d(z,Z) \le 2d(z,V)$.*

In this theorem, c is a universal constant that does not depend on f.

Fixed points of the Newton–Gauss iteration in the overdetermined case are not necessarily attractive. A similar constant γ can be defined, but this time the convergence (to a solution of the least-square problem $F'(x) = 0$ where $F(x) = \left\| f(x) \right\|^2$) is not quadratic anymore, see [3].

4. **Fast Deflation**

The results above do not apply in cases with multiple roots, or clusters of roots. In a slightly different context, that of finding series solutions of polynomial systems, G. Lecerf has devised in [6] an algorithm based on a Newton iteration with low complexity and quadratic convergence even in presence of multiplicities. We now give an outline of this algorithm and then sketch how this algorithm is adapted to the Archimedean world. To simplify notations, all series expansions are performed at the origin.

4.1. **Deflated System.** Let f be a system of n polynomial equations in n variables $x_1, \ldots, x_n$ having a finite number of solutions. To simplify the description, we assume some genericity conditions to be satisfied. The algorithm proceeds recursively by constructing an auxiliary block-triangular non-linear system (the deflated system). In each block, a subset of the variables $x_i, x_{i+1}, \ldots, x_{i+r_i-1}$ can be solved for in terms of the remaining ones $x_{i+r_i}, \ldots, x_n$ by means of a Newton iteration with quadratic convergence. At the end, it is therefore sufficient to solve these systems starting from the last one to get a solution of the original polynomial system.

The recursive step of this construction starts by computing the valuation of x_i (the first indeterminate which does not belong to one of the blocks constructed so far) in all the given equations. Let m_i be the smallest of these valuations. When computing series at a multiple root, $m_i > 1$ for $i > 1$ and the product of these m_i's is bounded by the multiplicity.

The equations are then differentiated j times with respect to x_i, for $j = 1, \ldots, m_i - 1$. A rectangular system Φ_i is formed with the original equations and these new equations. A subsystem Ω_i is then extracted, whose Jacobian has maximal possible rank r_i. By construction, Ω_i makes it possible to compute $x_i, \ldots, x_{i+r_i-1}$ by a Newton iteration with quadratic convergence, given values for the next variables. The next step is then started with Φ_i and the remaining variables as input.

4.2. **Series Expansions and Complexity.** Differentiation is needed at several stages during the algorithm: first when computing the systems Φ_i and then when performing Newton iterations on the systems Ω_i. However, the equations to be differentiated involve quantities that are known only implicitly as solutions of previous systems.

A way of computing the necessary derivatives is to compute multivariate series expansions at each stage. This means that each of the systems Ω_i is solved by a Newton iteration that converges both to the values and to the series expansions out of which extracting coefficients yields the values of the desired derivatives.

The nontrivial proof that this method leads to a correct algorithm with good complexity properties can be found in [6].

4.3. **Towards a Numerical Fast Deflation.** The idea of this work [4] is to follow the same steps as in Lecerf's algorithm, using numerical tools at intermediate steps.

The computation of valuations in series is replaced by the computation of multiplicity by the algorithm of Section 2.2. Because some of the variables are expressed as solutions of previous Ω_i's, the equations are not polynomial anymore. The idea is to work as if they were polynomial, using the Weierstrass preparation theorem to compute the required bounds. The computation of the rank is performed, for instance, by an LU-decomposition with a proper threshold to erase smaller diagonal entries. The computation of series is performed as in the symbolic case, with floating point coefficients that are themselves found by the Newton iteration.

What remains then is a rigorous analysis of how the convergence of this method and its quadratic behaviour can be related to the geometry of the problem via analogues of the γ-functions used in the theorems presented here. This will be treated in [4].

4.4. **Example.** We treat in detail a system borrowed from [6]: $f = g = h = 0$, with

$$f = 2x+2x^2+2y+2y^2+z^2-1, \quad g = (x+y-z-1)^3-x^3, \quad h = (2x^3+5y^2+10z+5z^2+5)^3-1000x^5.$$

This system has a solution of multiplicity 18 at $(0, 0, -1)$.

The computation begins with initial point $(.2, .1, -.98)$. Setting $y = .1$, $z = -.98$ in the system and using the algorithm of Section 2.2 reveals that .2 is close to a *simple* root for x, in the first equation only. Thus no differentiation is needed at this stage and the first block of the final system is $f = 0$. This defines a function $X(y, z)$ that can be computed by Newton iteration, as well as its Taylor expansion. The next step considers the remaining equations at $z = -.98$, x being replaced by $X(y, -.98)$. Applying the same technique shows that .1 is close to a root of multiplicity 3 for y in g and 5 in h. Thus the second block of the final system is formed by $\partial^2 g/\partial y^2\big(X(y, z), y, z\big)$ which defines a function $Y(z)$ that can be computed by Newton iteration, together with its Taylor expansion. Finally, $H(z) = \partial^2 h/\partial y^2\big(X(Y(z), z), Y(z), z\big)$ is found to have a root with multiplicity 4 for z close to $-.98$. This yields the last block of the system: $H^{(3)}(z) = 0$. Note that the product of the multiplicities that have been found—$1 \times 3 \times 4 = 12$—is smaller than the actual multiplicity 18.

One iteration of the algorithm on the system consists, for each of the blocks, in performing several Newton iterations to compute sufficiently many terms of the series expansion. After each such Newton iteration, the previous coordinates are updated using the derivatives of their series.

Thus, we start with f which we evaluate at $x = .2 + d_x$, $y = .1 + d_y$, $z = -.98 + d_z$. This yields

$$S(d_x, d_y, d_z) = (.6604 + 2.4d_y - 1.96d_z + 2d_y^2 + d_z^2) + 2.8d_x.$$

From there, the derivative with respect to x is obtained as the coefficient of d_x and we get as a result of the Newton iteration

$$X = .2 - S(0, d_y, d_z)/(\partial S/\partial d_x) = -.0358571 + .7d_z - .3571428d_z^2 - .8571429d_y - .7142857d_y^2.$$

Iterating again twice with $f(X + d_x, .1 + d_y, -.98 + d_z)$ yields

$$X = -.10022540 + 1.2248159 d_z - 2.4860889 d_z^2 + \cdots + O(d_y^4 + d_z^5).$$

We now turn to the second block of the deflated system, $\partial^2 g / \partial y^2 \big(X(y,z), y, z\big)$. To perform a Newton iteration on this block, g is first evaluated at $X, .1 + d_y, -.98 + d_z$ and then differentiated twice with respect to d_y. This yields

$$\begin{aligned} S(d_y, d_z) &= 1.5559281 - 31.251336 d_z + 269.81555 d_z^2 - 1962.5419 d_z^3 + 13304.819 d_z^4 \\ &\quad + (42.453768 - 708.11523 d_z + 7291.0178 d_z^2 - 63814.277 d_z^3 + 506494.98 d_z^4) d_y + O(d_y^2 + d_z^5). \end{aligned}$$

From there a Newton iteration yields

$$Y = .063350059 + .12481705 d_z + 2.0206612 d_z^2 + 3.4053518 d_z^3 + 21.246801 d_z^4 + O(d_z^5).$$

This value is then used to *update* the estimate X obtained at the previous stage, via

$$X_{\text{new}} := X_{\text{old}} + \partial X / \partial d_y (Y_{\text{new}} - Y_{\text{old}}).$$

The new estimate, $X = -.045258749 + .87056807 d_z - 4.1178532 d_z^2 + \cdots + O(d_y^4 + d_z^5)$, is then used together with Y to perform another Newton iteration on this block, and another update of X.

The treatment of the last block is similar. The only novelty is that the second derivative of h with respect to y has to be computed. This is achieved by evaluating h at $X(d_y, d_z), Y(d_z) + d_y, -.98 + d_z$, and extracting the coefficient of d_y^2. Then, differentiating three times with respect to d_z yields

$$S = 1673.8759 + 59921.515 d_z + O(d_z^2),$$

from which one Newton iteration gives $Z = -1.0079345$ and updating the previous coordinates

$$\begin{aligned} X &= -.0020677183 + .069477760 d_z + .44632326 d_z^2 + \cdots + O(d_y^4 + d_z^5), \\ Y &= .0023005230 + .85445586 d_z - 1.2958872 d_z^2 + 36.553123 d_z^3 - 250.55237 d_z^4 + O(d_z^5). \end{aligned}$$

In brief, this iteration of the algorithm has led from $(.2, .1, -.98)$ to $(-.0021, .0023, -1.008)$, from distance .2 to distance $8\,10^{-2}$ to the root. Similarly, the next iterations are respectively at distance of order $8\,10^{-4}$, $6\,10^{-6}$, $3\,10^{-9}$, $6\,10^{-18}$ and $2\,10^{-36}$ from the root, thus exhibiting a clearly quadratic behaviour after the first few iterations.

Bibliography

[1] Blum (Lenore), Cucker (Felipe), Shub (Michael), and Smale (Steve). – *Complexity and real computation.* – Springer-Verlag, New York, 1998, xvi+453p.

[2] Dedieu (Jean-Pierre). – Condition number analysis for sparse polynomial systems. In *Foundations of computational mathematics.* pp. 75–101. – Springer, Berlin, 1997. Proceedings of a conference held at Rio de Janeiro, 1997.

[3] Dedieu (Jean-Pierre). – Newton's method and some complexity aspects of the zero-finding problem. In DeVore (Ronald A.), Iserles (Arieh), and Süli (Endre) (editors), *Foundations of computational mathematics (Oxford, 1999)*, pp. 45–67. – Cambridge University Press, Cambridge, 2001. Proceedings of FoCM'99.

[4] Giusti (Marc), Lecerf (Grégoire), Salvy (Bruno), and Yakoubsohn (Jean-Claude). – Numerical Newton iteration with quadratic convergence to multiple solutions. – In preparation.

[5] Hubbard (John), Schleicher (Dierk), and Sutherland (Scott). – How to find all roots of complex polynomials by Newton's method. *Inventiones Mathematicae*, vol. 146, n° 1, 2001, pp. 1–33.

[6] Lecerf (Grégoire). – *Une alternative aux méthodes de réécriture pour la résolution des systèmes algébriques.* – Thèse de doctorat, École polytechnique, September 2001.

[7] Yakoubsohn (Jean-Claude). – Finding a cluster of zeros of univariate polynomials. *Journal of Complexity*, vol. 16, n° 3, 2000, pp. 603–638. – Complexity theory, real machines, and homotopy (Oxford, 1999).