Computing the Newtonian Graph

DEXTER KOZEN[†] AND KJARTAN STEFÁNSSON[‡]

[†] *Computer Science Department, Cornell University, Ithaca, NY 14853-7501, USA*
[‡] *Caliper Corporation, 1162 Beacon Street, Newton, MA 02161, USA*

(Received 5 August 1995)

In his study of Newton's root approximation method, Smale (1985) defined the Newtonian graph of a complex univariate polynomial f. The vertices of this graph are the roots of f and f' and the edges are the degenerate curves of flow of the Newtonian vector field $N_f(z) = -f(z)/f'(z)$. The embedded edges of this graph form the boundaries of root basins in Newton's root approximation method. The graph defines a treelike relation on the roots of f and f', similar to the linear order when f has only real roots.

We give an efficient algebraic algorithm based on cell decomposition to compute the Newtonian graph. The resulting structure can be used to query whether two points in $\mathbb{C}$ are in the same basin. This gives us a modified version of Newton's method in which one can test whether a step has crossed a basin boundary.

Stefánsson (1995) has recently extended this algorithm to handle rational and algebraic functions without significant increase in complexity. He has shown that the Newtonian graph tesselates the associated Riemann surface and can be used in conjunction with Euler's formula to give an *NC* algorithm to calculate the genus of an algebraic curve.

1. Introduction

Following Smale (1985), we define the *Newtonian vector field* of a polynomial $f \in \mathbb{C}[z]$ by $N_f(z) = -f(z)/f'(z)$. The name is derived from Newton's method for root approximation, in which successive approximations to a root of f are computed by the rule $x_{k+1} \leftarrow x_k + N_f(x_k)$.

The vector field N_f defines a flow on $\mathbb{C}$, where the flow comes almost everywhere from a pole (in the affine case where f is a polynomial, from ∞) and converges almost everywhere to a root of f. Each discrete step in Newton's method is tangent to a curve of flow. We can think of a curve of flow as the trajectory a particle would take under a version of Newton's method with infinitesimal steps. Certain degenerate curves of flow connect roots of f and f', and these degenerate curves form the boundaries of finitely many regions called *basins*, each containing a root of f. The *Newtonian graph* is a graph embedded in the complex plane whose vertices consist of the roots of f and f' and whose edges are these degenerate curves of flow. We define the graph more formally in §2. This graph has been studied and the possible graphs that can arise have been classified for polynomials by Shub *et al.* (1988).

In §3.2, we give a symbolic algorithm to compute a discrete model of the Newtonian graph of a given polynomial. The output of the algorithm is a labeled oriented graph that is topologically equivalent to the Newtonian graph, along with an oracle that can answer such questions such as

> Given $a, b \in \mathbb{C}$, are a and b in the same basin?
> Given $a, b \in \mathbb{C}$, are a and b on the same curve of flow?
> Given $a \in \mathbb{C}$, is a on a basin boundary?
> Given $a \in \mathbb{C}$, is a on an edge of the Newtonian graph?

Such a structure can be used in a version of Newton's method in which one can modify the step size at every step to ensure that we stay within a basin if desired.

2. The Newtonian Graph

We have defined the Newtonian vector field N_f of a complex univariate polynomial f. A vector field such as N_f on $\mathbb{C}$ defines a flow on $\mathbb{C}$. Given $c \in \mathbb{C}$, the flow through c is a function $\varphi_c : I \to \mathbb{C}$, where I is a real interval containing zero and φ_c differentiable with

$$\begin{aligned} \frac{d\varphi_c(t)}{dt} &= N_f(\varphi_c(t)) \\ \varphi_c(0) &= c \ . \end{aligned}$$

That is, φ parameterizes the flow starting at c, and at every point the direction of flow is tangent to the field. An illustration of the Newtonian vector field of a polynomial f of degree four is given in Figure 1.

The flow exists on all of $\mathbb{C} \backslash V_{f'}$ (where $V_{f'} = \{z \in \mathbb{C} \mid f'(z) = 0\}$). The existence and uniqueness follows from the theory of differential equations and the fact that N_f is a C^1 function on $\mathbb{C} \backslash V_{f'}$ (see *e.g.* Hirsch and Smale (1974), §8.2 and §8.5).

The following lemma of Shub *et al.* (1988) gives us an important property of the flow:

LEMMA 2.1. *Let $f \in \mathbb{C}[z]$ and let φ_c be the flow through c in the Newtonian field N_f. Then f maps the curve $\{\varphi_c(t) \mid t \in I\}$ to a ray pointing to the origin. More specifically,*

$$f(\varphi_c(t)) = f(c)e^{-t} \ .$$

PROOF. Computing $df(\varphi_c(t))/dt$ using the chain rule gives:

$$\begin{aligned} \frac{df(\varphi_c(t))}{dt} &= f'(\varphi_c(t))\frac{d\varphi_c(t)}{dt} \\ &= f'(\varphi_c(t))N_f(\varphi_c(t)) \\ &= -f(\varphi_c(t)) \ , \end{aligned}$$

which is a differential equation in t for the function $f \circ \varphi_c$. Given the initial condition $\varphi_c(0) = c$, it has the unique solution $f(\varphi_c(t)) = f(c)e^{-t}$. □

One consequence of Lemma 2.1 is that the flow functions $\varphi_c(t)$ are algebraic over $\mathbb{C}[e^t]$. Using the properties of φ, one can show the following.

LEMMA 2.2. *For every $c \in \mathbb{C} \backslash (V_f \cup V_{f'})$, φ_c is defined on a maximal real interval (a, b) containing 0, which is of one of the following four types:*

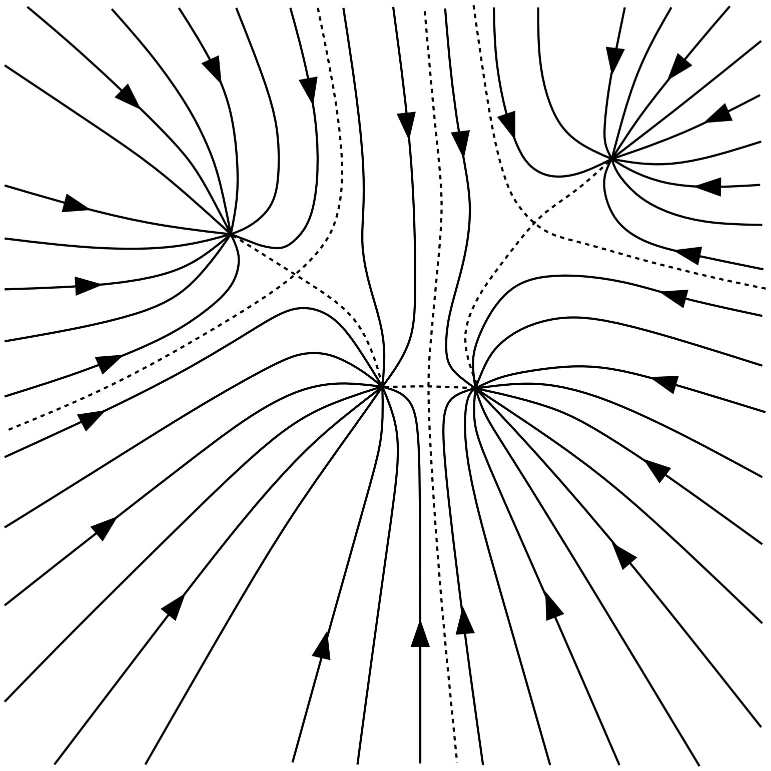

Figure 1. The Newtonian vector field of a polynomial of degree four. Every curve of flow is directed to a root, except the basin boundaries (dotted lines). There is a root of f' on every basin boundary, and a curve of flow from there to "adjacent" roots (also dotted lines).

1 $(-\infty, +\infty)$*, and the flow comes from* ∞ *and goes to a root of* f*;*
2 $(-\infty, b)$ *and the flow comes from* ∞ *and goes to a root of* f'*;*
3 (a, b) *and the flow comes from a root of* f' *and goes to another root of* f'*;*
4 $(a, +\infty)$ *and the flow comes from a root of* f' *and goes to a root of* f*.*

PROOF. N_f is a C^1 function $W \to \mathbb{C}$ where $W = \mathbb{C} - (V_f \cup V_{f'})$, and all flow must leave any compact set of W (Hirsch and Smale (1974), §8.5). By Lemma 2.1, the maximal interval of φ_c is unbounded upwards iff the flow goes to a root of f. The same argument shows that the interval is not bounded below iff the flow comes in from ∞. Since the flow leaves any compact set of W, the only other limit points are in $V_{f'}$. $\square$

DEFINITION 2.1. *The* Newtonian graph *of a polynomial* $f \in \mathbb{C}[z]$ *is the embedded plane graph* $G = (V, E, \pi)$ *with vertices* $V = V_f \cup V_{f'}$*, directed edges consisting of the curves of flow between vertices wherever they exist, and orientation* π *of edges about any vertex determined by the embedding.*

We note that the graph is not just a combinatorial structure, but also includes an orientation as determined by the embedding.

Figure 2 illustrates the Newtonian graph of the vector field of Figure 1. Figure 3 shows an example containing an edge between two roots of f'.

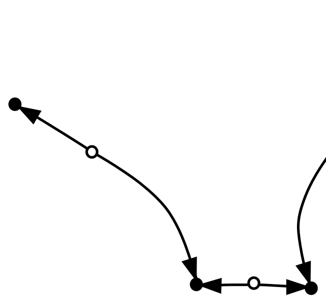

Figure 2. The graph of the vector field in Figure 1. The solid dots are the roots of the polynomial, the hollow ones are the roots of the derivative.

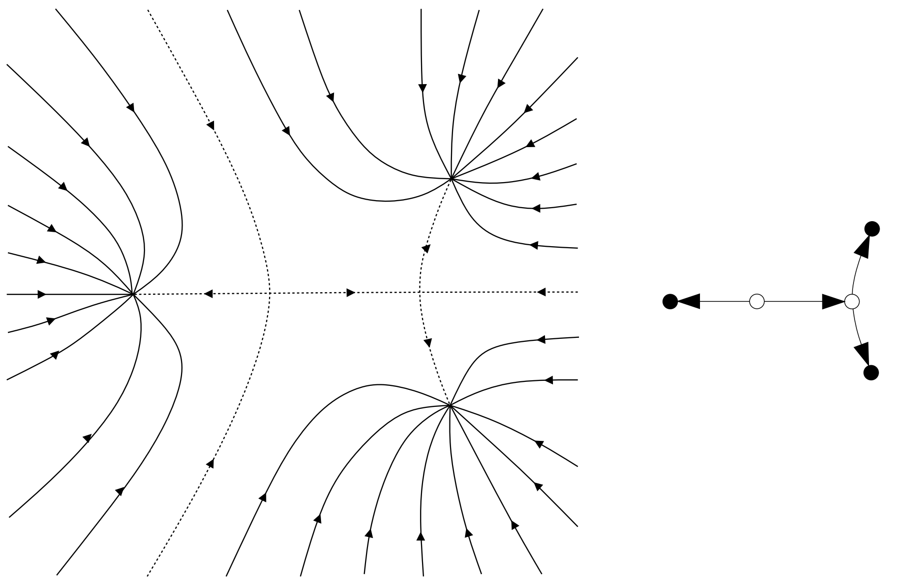

Figure 3. The vector field and the graph of a degree three (real) polynomial with an edge between two derivative roots.

Under f, every edge maps onto a line segment of finite length with endpoints in the set $\{f(c) \mid f'(c) = 0\} \cup \{0\}$ and lying on a ray through the origin. This is an immediate consequence of Lemma 2.1 and the fact that edges are curves of flow. Conversely, the preimage under f of any such ray consists of only finitely many curves, at most the degree of f. Thus the graph has only finitely many edges. Furthermore, Shub *et al.* (1988) show that the graph is connected, and classify the possible types of graphs that can arise.

A *basin of attraction* is a connected region consisting of flow going to one particular root of f. A *basin boundary* is the boundary between two basins. There must be a root of f' on every basin boundary, because flows are continuous, and it requires a discontinuity of N_f for the flow to "split" into two directions, and these only occur at the roots of f'. Also the basin boundaries are curves of flow themselves, so we conclude that every basin boundary is flow into a root of f'. In particular this means that basin boundaries are contained in the preimage $f^{-1}(\{mf(c) \mid f'(c) = 0,\ 1 \leq m\})$.

3. Computing Basins and Graph Edges

We will give an algorithm to compute the basin boundaries and the edges of the Newtonian graph. First we need a few preliminaries on cylindric algebraic decomposition.

3.1. Cell Decomposition

We describe cylindric algebraic cell decomposition briefly. For a more detailed description, see Collins (1975) or Ben-Or *et al.* (1986).

Definition 3.1. *A* decomposition *of $\mathbb{R}^k$ is a finite partition $\{C_i\}_{i\in I}$ such that each C_i is connected, $C_i \cap C_j = \varnothing$ if $i \neq j$, and $\bigcup_{i\in I} C_i = \mathbb{R}^k$. For $k = 1$, such a decomposition is* cylindric *if the each C_i is either a point or an interval. For $k > 1$, the decomposition is cylindric if for all r, $1 \leq r \leq k$, $\{\pi_{1,\ldots,r}(C_i) \mid i \in I\}$ is a cylindric decomposition of $\mathbb{R}^r$.*

Definition 3.2. *Given polynomial equations, $f_i(x_1,\ldots,x_m) = 0$, $1 \leq i \leq n$, with $f_i \in \mathbb{R}[x_1,\ldots,x_m]$, a* cylindric algebraic decomposition *(CAD) of $\mathbb{R}^m$ is a data structure $\mathcal{D}$ with the following properties.*

$\mathcal{D}$ contains a graph whose nodes correspond to certain subsets of $\mathbb{R}^m$ called cells*, each cell homeomorphic to $\mathbb{R}^d$ for some d, and the cells are a decomposition of $\mathbb{R}^m$.*
For all $i = 1,\ldots,n$, $\mathrm{sign}(f_i)$ is constant on every cell. Each cell is labeled with the signs that the f_i take on that cell.
Every node contains an oracle such that given any $c \in \mathbb{R}^m$, the oracle can answer if c is contained in the associated cell.
Every node contains dimension information, corresponding to the dimension of the associated cell.
The edges of the graph correspond to adjacency of the cells in $\mathbb{R}^m$. There is a directed edge (u, v) if the cell associated with u forms part of the boundary of the cell associated with v.
The decomposition is cylindric.

Algorithms have been developed to compute (parts of) such a cell decomposition dating back to Tarski (1951). Collins (1975) has a double exponential algorithm, although it lacks some of the adjacency information. Ben-Or *et al.* (1986) developed a parallel algorithm giving the same kind of decomposition (the BKR algorithm), and Kozen and Yap (1985) extended that algorithm to obtain full adjacency information as well (hereafter called the extended BKR algorithm).

We note that due to the cylindric condition and adjacency information, an algorithm computing such a decomposition can be used on a set of polynomials with quantifiers, projecting down the result. If the input is a formula of the form

$$\exists y_1 \exists y_2 \ldots \exists y_k \bigwedge_{i=1}^{n} f_i(x_1,\ldots,x_m,y_1,\ldots,y_k) \quad = \quad 0\ ,$$

we can perform a CAD on $\mathbb{R}^{m+k}$, then project the solution down to $\mathbb{R}^m$. The resulting structure can be used to answer questions of the form: Given $c \in \mathbb{R}^m$, does there exist $y_1,\ldots,y_k \in \mathbb{R}^k$ such that $y_1,\ldots,y_k, c$ is a solution to the system?

We note that the order of variables is important with respect to the cylindric condition.

3.2. The Algorithm

Recall that every basin boundary and every edge is mapped by f onto a straight line. Also, the basin boundaries and edges have a root of f' as a limit. Thus, all these "interesting" curves of flow satisfy, for every z on the curve,

$$\exists c \in \mathbb{C}\ \exists m \in \mathbb{R}\ f(z) = mf(c) \quad \wedge \quad f'(c) = 0 \ . \tag{3.1}$$

Any point z on a basin boundary or an edge must satisfy these two conditions. We note that the converse is false in general; $z \in \mathbb{C}$ can be a solution to (3.1) without being on an edge or a basin boundary.

We proceed in two steps. First we find a decomposition of $\mathbb{C}$ describing the solutions z to (3.1). Then we prune that output, because we may get spurious solution curves that do not correspond to basin boundaries or edges.

To find the solutions to (3.1), we compute a cylindric algebraic decomposition based on the equations

$$f(z) = mf(c) \qquad\qquad f'(c) = 0 \ . \tag{3.2}$$

The resulting structure gives a decomposition of $\mathbb{R} \times \mathbb{C} \times \mathbb{C}$ describing regions where such m, c, z exist, along with the dimension of each region and adjacency information. Projecting m and c, we obtain curves in $\mathbb{C}$ for which there exists a solution to (3.1).

First let us note that algorithms such as Collins' and the extended BKR algorithm do decomposition over the reals. But we can split the equations into a real and imaginary parts, and get a decomposition of $\mathbb{R}^5 \cong \mathbb{R} \times \mathbb{C} \times \mathbb{C}$, corresponding to the equations

$$\begin{aligned} f_R(x,y) &= mf_R(c_1,c_2) \\ f_I(x,y) &= mf_I(c_1,c_2) \\ f'_R(c_1,c_2) &= 0 \\ f'_I(c_1,c_2) &= 0 \ , \end{aligned}$$

where $f(x+iy) = f_R(x,y) + if_I(x,y)$ and $f_R, f_I \in \mathbb{R}[x,y]$. We get a decomposition on $\mathbb{R}^5$ which corresponds to a decomposition on $\mathbb{R} \times \mathbb{C} \times \mathbb{C}$.

We then project the dimensions corresponding to $c = c_1 + ic_2$ and again project m, obtaining a decomposition of $\mathbb{C}$ corresponding to z for which there exist m and c satisfying equation (3.2).

This decomposition will contain all the basin boundaries and graph edges. These will be partitioned into segments (bounded 1-cells) and 0-cells between such segments. There may be other cells present which are not part of solutions to the system (3.2), corresponding to auxiliary cells introduced by the CAD algorithm. However, we can always identify these, because all cells are labeled with the signs of the input polynomials (3.2), which determine which of them constitute actual solutions. A solution curve to the system can be reconstructed by linking such adjacent cells.

Not all solution curves are edges or basin boundaries. The following lemma classifies the types:

Lemma 3.1. *The output from the process above contains at most $O(n^2)$ 1-cells which are solutions curves for the input system. They are of the following types:*

(i) Adjacent to two 0-cells, one of which describes a root of f' and one which describes a root of either f or f';

(ii) Adjacent only to one 0-cell which describes a root of f';
(iii) Adjacent only to one 0-cell which describes a root of f.

Cells of type (i) are edges of the Newtonian graph; cells of type (ii) are a basin boundaries; and cells of type (iii) are extraneous solutions to the system.

Proof. The only 1-cells that can be solutions to the system correspond to curves of flow. The classification is obvious from the definition of edges and the properties of basin boundaries.

There are at most $O(n^2)$ solution curves for $f(z) = mf(c)$ with c a root of $f'(c)$ and $m \in \mathbb{R}$, because there are at most $n-1$ roots of f' and f is an n-to-one mapping. $\square$

The cells of type (i) and (ii) are the ones we are interested in. We can distinguish these from the extraneous cells of type (iii) by checking the sign of f, which allows us to verify if a curve ends at a root of f. Since f is part of the input, the sign of f is available on every cell.

Depending on which algorithm we use, we may or may not have all the information needed. The extended BKR guarantees that if f is a part of the input, then the signs of f' will be provided on each cell. If we do not have this guarantee, we can always add $f'(z) = 0$ to our input equations and get the same information.

At this point we can determine the types of the solution curves. Now it is easy to implement the pruning step: we simply coalesce each cell of type (iii) as part of the adjacent 2-cell, which is the basin that this cell lies in.

Now the structure can be used in answering queries. Two points are in the same basin if they are in the same 2-cell or if they are separated only by 1-cells of type (iii).

4. Improvements

Recall that we computed a CAD of $\mathbb{R}^5 \cong \mathbb{R} \times \mathbb{C}^2$ with respect to the equations

$$f(z) = mf(c) \qquad f'(c) = 0$$

and projected the solution onto $\mathbb{C}$. This can be simplified by defining

$$g(m, z) \quad = \quad \mathrm{Res}_c(f(z) - mf(c), f'(c)),$$

where Res_c denotes the univariate resultant with respect to c. Here we view $f(z) - mf(c)$ and $f'(c)$ as univariate polynomials in $\mathbb{C}[z, m][c]$.

The polynomial g has the property that $g(m, z) = 0$ iff $\exists c \in \mathbb{C}\ f(z) - mf(c) = 0 = f'(c)$. Hence, a decomposition of $\mathbb{R} \times \mathbb{C}$ with respect to g is the same as the projection of the decomposition of $\mathbb{R} \times \mathbb{C} \times \mathbb{C}$ with respect to the original two equations.

The only thing we must be aware of is how to obtain the necessary signs of f and f' on cells, in order to identify and link up solution curves and prune off the spurious ones. One way would be to add the equation $f(z) = 0$ (and $f'(z) = 0$, if we are not using the extended BKR), and do a decomposition with respect to f (f') and g. This is already an improvement in terms of dimension, since we are only working with three real variables ($x = \Re z$, $y = \Im z$ and m) instead of five.

The asymptotic complexity remains the same, but the constants are much better. The extended BKR gives an NC circuit of depth $2^{O(d^2)} \log^{O(d)} n$ where d is the number of variables and n is the maximum of either the number of polynomials or their degrees.

In our case the circuit will be of depth $O(\log^{O(1)} n)$ where n is the degree of the input polynomial f.

5. Applications to Newton's Method

The ability to test whether two points lie in the same basin of the Newtonian vector field opens up intriguing possibilities for Newton's root approximation method. Since one can test whether a Newton iteration step has jumped over a basin boundary, one can modify the algorithm to scale back the step size to stay within a particular basin if desired. This can be done for example by replacing the usual Newton step $z_{k+1} \leftarrow z_k + N_f(z_k)$ by the program

```
$\alpha \leftarrow 1$,
**repeat**
  $z_{k+1} \leftarrow z_k + \alpha N_f(z_k)$
  $\alpha \leftarrow \alpha/2$
**until** $z_{k+1}$ is in the same basin as $z_k$
```

Here we use our precomputed Newtonian graph structure to determine whether two points are in the same basin. If in addition strict progress toward a root is desired, one can modify the last line of the program to read

```
**until** $z_{k+1}$ is in the same basin as $z_k$ and $|z_{k+1}| < |z_k|$
```

One might conjecture that this approach gives a version of Newton's method in which convergence to a root is guaranteed. Unfortunately, this is not the case, as shown by the following counterexample.

Consider a polynomial f with a basin boundary φ_c such that φ_c has strictly positive curvature and f, f', and f'' do not vanish in a neighborhood of $c = \varphi_c(0)$.

For instance, we might take $f(z) = z^3 - z$ with roots $-1, 0, 1$, derivative roots $\pm 1/\sqrt{3}$, and basin boundary φ_c with c the unique root of $c^3 - c + 2\sqrt{3}e/9$ in the positive quadrant. In this case

$$f(\varphi_c(t)) \quad = \quad f(c)e^{-t} \quad = \quad -\frac{2\sqrt{3}}{9}e^{1-t} \ .$$

It follows that for $t < 1$, $\varphi_c(t)$ is the unique root of $x^3 - x + 2\sqrt{3}e^{1-t}/9$ in the positive quadrant, and $\varphi_c(1) = 1/\sqrt{3}$, thus φ_c is a basin boundary. This example is illustrated in Figure 4.

Let N be an open ball about c of sufficiently small radius. Let A be the portion of N to the left of φ, moving along φ in the direction of positive t, and let B be the portion of N to the right of φ (in Figure 4, A appears to the right of φ). By the assumption about the curvature of φ, $(A \cup \varphi) \cap N$ is a convex set. Also, the radius of N can be chosen small enough that all flow lines have strictly positive curvature in N.

We will consider scaled Newton steps $z \mapsto z + \epsilon N_f(z)$ applied to $z \in A$. Our modified Newton's method described above, applied to a point $z \in A$, gives

$$MN(z) \quad = \quad z + 2^{-k}N_f(z)$$

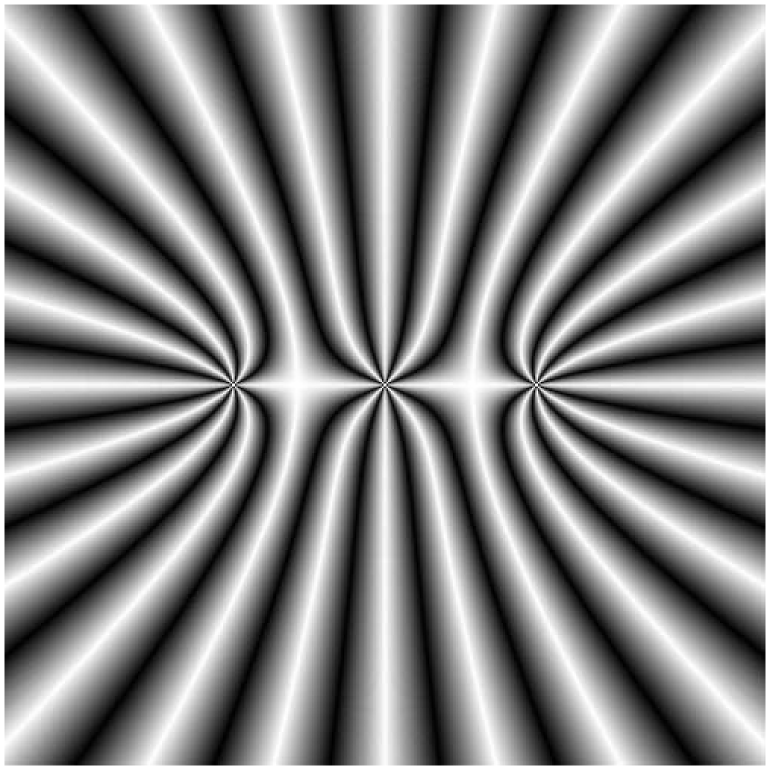

Figure 4. A counterexample.

where k is the least nonnegative integer such that $z + 2^{-k}N_f(z) \in A$. (For this counterexample, the extra test $|z + 2^{-k}N_f(z)| < |z|$ is rendered superfluous by picking the radius of N sufficiently small.)

Let a be a point in A such that the line segment ac is perpendicular to $N_f(c)$. For t in the real interval $[0, 1]$, define $u(t) = ct + (1-t)a$. The function $u(t)$ travels along the segment ac as t goes from 0 to 1. By convexity, all points $u(t)$ lie in A except for the endpoint $u(1)$, which lies on φ.

We will construct a *scaling sequence*

$$\epsilon_0 > \epsilon_1 > \epsilon_2 > \quad \cdots \quad > 0$$

of small positive reals such that each $\epsilon_i = 2^{-k_i}$ for some positive integer k_i, and the ϵ_i converge sufficiently rapidly to 0 to satisfy several conditions given below. Relative to the scaling sequence $\{\epsilon_i\}$, we define the functions $u_n : [0, 1] \to \mathbb{C}$ by

$$\begin{aligned} u_0(t) &= u(t) \\ u_{n+1}(t) &= u_n(t) + \epsilon_n N_f(u_n(t)) \ . \end{aligned}$$

Thus $u_n(t)$ is the point obtained by starting from $u(t)$ and applying n scaled Newton iterations with scale factor ϵ_i at the ith step. Note that u_n depends on the scaling sequence $\{\epsilon_i\}$, although this dependence is not explicit in the notation.

One of the conditions on $\{\epsilon_i\}$ is that $u_n(t) \in N$ for all $t \in [0,1]$ and $n \geq 0$. Since

$$|u_{n+1}(t) - u_n(t)| \quad = \quad \epsilon_n |N_f(u_n(t))| \ ,$$

we can insure this by choosing ϵ_n sufficiently small, as follows. If $u_n(t) \in N$ for all $t \in [0,1]$, let $r(t)$ be the maximum radius of an open ball centered at $u_n(t)$ and wholly contained in N. The function $r : [0,1] \to \mathbb{R}$ is continuous and defined on a compact set, thus achieves its infimum $\inf_t r(t) > 0$ at some $t \in [0,1]$. We can inductively insure that $u_{n+1}(t) \in N$ for all $t \in [0,1]$ by picking

$$\epsilon_n \quad < \quad \frac{\inf_{t \in [0,1]} r(t)}{\sup_{z \in N} |N_f(z)|} \ .$$

We can also choose $\{\epsilon_i\}$ such that $u_n : [0,1] \to \mathbb{C}$ is one-to-one and $du_n(t)/du$ is arbitrarily close to 1 uniformly in t. The latter condition implies the former. A straightforward calculation gives

$$\frac{du_n}{du} \quad = \quad \prod_{i=0}^{n-1} (1 + \epsilon_i(\frac{f(u_i)f''(u_i)}{f'(u_i)^2} - 1)) \qquad (5.1)$$

and we have already insured that all $u_i(t) \in N$, therefore $f(u_i)f''(u_i)/f'(u_i)^2$ is bounded. We can thus choose the ϵ_n sufficiently small that the sequences $du_n(t)/du$ for $t \in [0,1]$ converge uniformly to values arbitrarily close to 1.

Intuitively, these conditions say that the locus of points $u_n(t)$ is nearly a straight line segment in N and nearly parallel to $u(t)$. In particular, it intersects φ at most once, and if it intersects, then it does so transversally (we chose the radius of N sufficiently small that the direction of $N_f(z)$ does not vary much).

Now we construct inductively two real sequences

$$0 = s_0 < s_1 < s_2 < \quad \cdots \quad < t_2 < t_1 < t_0 = 1$$

such that $u_n(t) \in A$ for $s_n \leq t < t_n$ and $u_n(t_n) \in \varphi$. This is already true for $n = 0$. Suppose we have constructed s_n and t_n. By the curvature assumption, $u_{n+1}(t_n) \in B$, and we can insure $u_{n+1}(s_n) \in A$ by having picked ϵ_n sufficiently small. Therefore there must exist a point t_{n+1} such that $s_n < t_{n+1} < t_n$ and $u_{n+1}(t_{n+1}) \in \varphi$. By the bound on du_{n+1}/du, we have that t_{n+1} is unique, $u_{n+1}(t) \in A$ for all $t < t_{n+1}$, and $u_{n+1}(t) \in B$ for all $t > t_{n+1}$.

Now $u_n(t_{n+1}) \in A$ and

$$u_{n+1}(t_{n+1}) = u_n(t_{n+1}) + \epsilon_n N_f(u_n(t_{n+1})) \quad \in \quad \varphi \ ,$$

so by the curvature assumption,

$$u_n(t_{n+1}) + 2\epsilon_n N_f(u_n(t_{n+1})) \quad \in \quad B \ .$$

By choosing $s_{n+1} \in (s_n, t_{n+1})$ sufficiently close to t_{n+1}, we can insure

$$\begin{aligned} u_{n+1}(t) = u_n(t) + \epsilon_n N_f(u_n(t)) \quad &\in \quad A \\ u_n(t) + 2\epsilon_n N_f(u_n(t)) \quad &\in \quad B \end{aligned}$$

for all $t \in (s_{n+1}, t_{n+1})$. This says that our modified Newton step gives

$$MN(u_n(t)) \quad = \quad u_{n+1}(t) \qquad (5.2)$$

for all $t \in (s_{n+1}, t_{n+1})$.

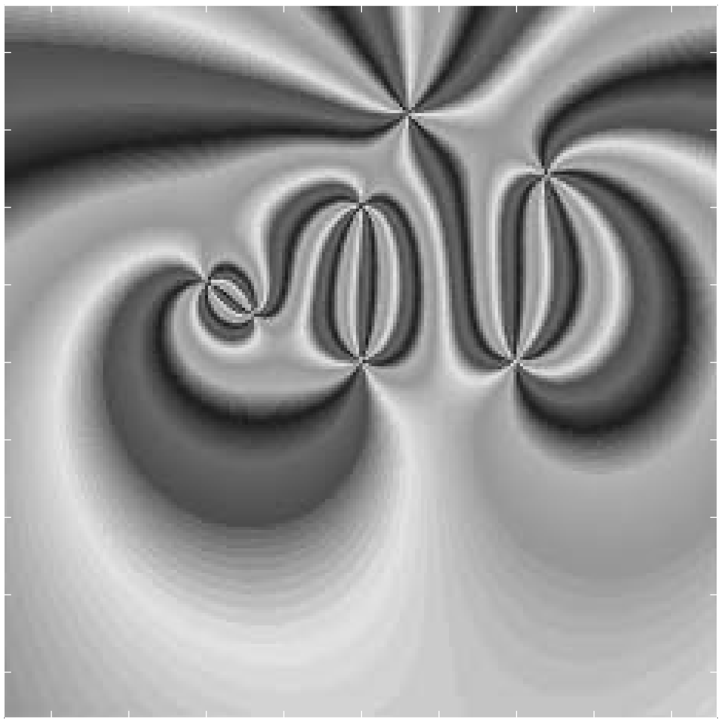

Figure 5. The Newtonian vector field of a rational function.

Now let $t_* = \inf_n t_n \geq \sup_n s_n$. By (5.2), we have that

$$MN(u_n(t_*)) \quad = \quad u_{n+1}(t_*) \ .$$

Thus the modified Newton algorithm started at $u(t_*)$ converges to a point on φ in the closure of N, far from a root of f.

Despite the failure of the modified method to converge to a root in all cases, the ability to test membership in a particular basin raises other intriguing possibilities. For example, one might test whether one is very close to a basin boundary by counting the number of times the step size was halved, and take a "sideways" step toward the interior of the basin if so. Such modifications present themselves as interesting topics for future investigation.

6. Newtonian Graphs of Rational and Algebraic Functions

Stefánsson (1995) has extended the definition of the Newtonian graph to rational and algebraic functions and has extended the algorithm of §3.2 to handle these more general cases with no significant increase in complexity.

Figure 5 illustrates the Newtonian vector field of a complex rational function of degree four. Three poles and four roots are visible; there is a fourth pole at ∞. Curves of fixed color indicate curves of flow.

Stefánsson (1995) has shown that for rational and algebraic functions, the Newtonian graph tesselates the associated Riemann surface, and in conjuction with Euler's formula gives an NC algorithm to calculate the genus of the surface.

7. Acknowledgments

The support of the National Science Foundation under grant CCR-9317320 and the U.S. Army Research Office through the ACSyAM branch of the Mathematical Sciences Institute of Cornell University under contract DAAL03-91-C-0027 is gratefully acknowledged.

References

Ben-Or, M., Kozen, D., Reif, J. (1986). The complexity of elementary algebra and geometry. *J. Comput. Syst. Sci.* **32:2**, 251–264.

Collins, G.E. (1975). Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Lect. Notes in Comput. Sci.* **33**, 134–183. Springer-Verlag.

Hirsch, M.W., Smale, S. (1974). *Differential Equations, Dynamical Systems, and Linear Algebra.* Academic Press.

Kozen, D., Yap, C.K. (1985). Algebraic cell decomposition in *NC*. In *Proc. 26th Symp. Found. Comput. Sci.*, 515–521. IEEE.

Shub, M., Tischler, D., William, R.F. (1988). The Newtonian graph of a complex polynomial. *SIAM J. Math. Anal.* **19:1**.

Smale, S. (1985). On the efficiency of algorithms of analysis. *Bull. Amer. Math. Soc.* **13:2**, 87–122.

Stefánsson, K. (1995). *Newtonian Graphs, Riemann Surfaces, and Computation.* PhD thesis, Cornell University.

Tarski, A. (1951). *A decision method for elementary algebra and geometry.* Univ. of Calif. Press, second edition.