The Numerical Inversion of Functions from the Plane to the Plane

Iaci Malta, Nicolau C. Saldanha, Carlos Tomei

Abstract. This paper contains a description of a program designed to find all the solutions of systems of two real equations in two real unknowns which uses detailed information about the critical set of the associated function from the plane to the plane. It turns out that the critical set and its image are highly structured, and this is employed in their numerical computation. The conceptual background and details of implementation are presented. The most important features of the program are the ability to provide global information about the function and the robustness derived from such topological information.

1991 *Mathematics Subject Classification.* Primary 57R45, 65H10; Secondary 57R42, 65H20.

Introduction

In this paper, we describe a theoretical framework, and a program based thereupon, designed to solve systems of two real equations in two real unknowns,

$$F(x) = a, \qquad x, a \in \mathbb{R}^2, \tag{1}$$

for *nice* functions F, i.e., generic proper smooth functions with bounded critical set in a sense to be made precise in § 1. The program is reasonably fast, but its most important features are the ability to provide global information about the function and the robustness derived from such topological information. Most of the program consists in computing the critical set C of F: as we shall see, the sets C and $F(C)$ are highly structured. In particular, the number of solutions of (1) is determined from knowledge of C and $F(C)$. With $F(C)$ available, the situation is perfect for the use of continuation methods, as will be seen in § 3. We thus need criteria to decide if a computed collection $C_\bullet$ of critical curves may possibly be the entire critical set, in which case we may safely begin the inversion procedure. An application of the inversion algorithm is the computation of the set $\mathcal{F} = F^{-1}(F(C))$, the *flower* of F, from which one obtains visual understanding of the function.

We make extensive use of standard techniques in nonlinear numerical analysis: excellent references are [1] and [6]. The additional structure of our problems, however, allows our algorithm to be more complete than general equation-solving procedures, such as global Newton methods ([5]). On the other hand, unlike solving complex polynomial systems ([10]), our problems are not amenable to an algebraic approach. We assume as given routines which return the values of F and its first derivatives at any given point, which is obviously insufficient for any algorithm to fully and reliably solve our problems. Throughout the paper, we indicate some of the steps where further, a priori knowledge of F would be necessary to guarantee faultless performance of the suggested algorithms.

In §1, we list the conditions satisfied by the critical set C and its image $F(C)$. The simplest are counting relations, given in Proposition 1.3, which play the role of the Riemann-Hurwitz theorem in our context. Then, in Theorem 1.6, we present a combinatorial *word criterion* which allows us to decide if a given set $C_\bullet$ is indeed the critical set of a nice function with specified behavior in its neighborhood. The criterion relies on previous work by Blank and Troyer (Theorem 1.4; see [11] and [13]) on immersions of disks with holes in the plane.

In §2, we discuss the computation of the critical set. In a nutshell, the program verifies if the conditions above are satisfied by the set $C_\bullet$. If counting relations, which are easier to check, are satisfied, the program checks the word criterion. A convenient feature of both kinds of conditions is that, if they are not satisfied, they indicate where to search for another critical curve. The counting relations are rather stringent: a set $C_\bullet$ which satisfies them almost always complies with the word criterion.

In §3, we describe the algorithm employed to solve (1), given C and $F(C)$. We begin by computing all pre-images of one point p_0 somewhat removed from $F(C)$. For other arbitrary points p, we obtain a convenient path joining p_0 to p and perform inversion by continuation along the path. Notice that we know exactly where and how regular continuation breaks down.

In the last section, we provide a collection of examples. For a somewhat complicated function, the program computed its critical set C and then inverted a random collection of points scattered over a region containing parts of $F(C)$: this rather hard inversion problem turned out to be less than six times slower than the far easier problem of inverting another sample of points in a very regular region of the image of the same function. In another run, starting from an incomplete $C_\bullet$, the program, after checking counting relations, searched for additional critical curves until the whole set C had been found. Next, we illustrate the use of the flower to understand the global behavior of two simple functions. Another example shows that the program performs well when asked to invert points close to the images of folds or cusps. Finally, we consider a far more complicated function, with critical set consisting of 17 curves, and show that again inversion works well for nontrivial points.

1. Geometry of nice functions

The global geometry of a nice function F from the plane to itself is described in this section by making use of its critical sets C and $F(C)$ and of the pre-image $\mathcal{F} = F^{-1}(F(C))$. Theorem 1.6 is a necessary and sufficient condition for a known set of critical curves $C_\bullet$ of F to be the full critical set of a nice function coinciding with F in a neighborhood of $C_\bullet$.

We begin with some definitions, most of them standard. A continuous function F from the plane to itself is *proper* if the pre-image of any compact set in the plane is compact. A point in the domain of a differentiable function F is *regular* if the derivative DF at this point is invertible, and *critical* otherwise. The image of the set C of critical points is the set $F(C)$ of *critical values*; its complement in the plane is the set of *regular values*. Smooth proper functions F have a *topological degree* $\deg(F)$ (see [9]): the number

of pre-images of a regular value counted with the sign of the determinant of the Jacobian at each pre-image. A critical point x of F is a *fold point* (resp., a *cusp point*) if there are local orientation-preserving diffeomorphisms around x and $F(x)$ onto neighborhoods of the origin of the plane in which F takes the form (2) (resp. (3)) below:

$$F(u,v) = (u, v^2), \tag{2}$$

$$F(u,v) = (u, av^3 - uv), \quad a = \pm 1. \tag{3}$$

By a celebrated theorem of Whitney ([14]), in an appropriate topology, generic functions only have folds and cusps as critical points. A point of $F(C)$ is an *intersection point* if it has more than one critical pre-image. A *transversal double point* is an intersection point with exactly two critical pre-images, both of which are folds, and such that the tangent lines to the images of neighborhoods in C of the folds are distinct. Generically, smooth functions from the plane to the plane only have transversal double intersection points.

In this paper, we only consider *nice* functions F, defined by the requirements below.

(a) F is a smooth proper function from the plane into itself.

(b) The critical set C of F is bounded and each critical point is either a fold or a cusp.

(c) F only has transversal double intersection points.

For a nice F, $C = \bigcup \gamma_i$ is a finite disjoint union of simple closed regular curves γ_i with finitely many cusps. Also, except for images of cusps, $F(\gamma_i)$ is a smooth curve in the plane. Finally, $\deg(F) \neq 0$ and, in a neighborhood of infinity, a nice function F behaves topologically like the function $z \mapsto z^n$ for $n = \deg(F) > 0$ or $z \mapsto \overline{z}^n$ for $n = -\deg(F) > 0$, in complex notation.

We define the *sense of folding* of the critical curve γ in the domain to be the orientation which leaves neighboring points p with $\det(DF(p)) > 0$ to the left of γ. This induces by F an orientation on $F(\gamma)$, likewise called the sense of folding of $F(\gamma)$, with the following property: the image of a small neighborhood of a fold point $p \in \gamma$ lies entirely to the left of $F(\gamma)$. For a critical curve, the sense of folding corresponds to the positive (counterclockwise) orientation if and only if $\det(DF)$ is positive (immediately) inside the curve.

Let γ be a critical curve of a nice function F, bounding an open disk D. A cusp point $x \in \gamma$ will be called *inward* if, for each sufficiently small neighborhood U_x of x, $F^{-1}(F(\gamma)) \cap U_x$ is contained in $\overline{D}$; otherwise, it will be called *outward*. From the local form, x is inward if and only if $a \det(DF)|_{U_x \cap D} > 0$.

Given a nice function F, suppose we know

(1) a set $C_\bullet$ of critical curves,

(2) their images, $F(C_\bullet)$,

(3) the cusps in $C_\bullet$, together with their classification as inward or outward,

(4) the sense of folding on each curve in $F(C_\bullet)$,

(5) the value of $\deg(F)$.

The question we address is: is $C = C_\bullet$? In terms of these data, the best that can be done is to provide, as in Theorem 1.6, necessary and sufficient conditions for the existence of a nice function $F_\bullet$ with the following properties: $F_\bullet$ has the same topological degree as F, its critical set is $C_\bullet$, and it coincides with F in an open neighborhood of $C_\bullet$.

The next two propositions build up towards the first set of necessary conditions, given in Proposition 1.3. Any (open) connected component of the set of regular values of a nice function F will be called a *tile for* $F(C)$. From properness and the fact that F is a local homeomorphism outside the critical set, the number of solutions of $F(x) = a$ is constant for a in a tile for $F(C)$.

Proposition 1.1. *Any nice function F is surjective. If T is a tile for $F(C)$ and S is the pre-image of T, then $F : S \to T$ is a finite covering. Moreover, if T_∞ is the unbounded tile, then $F^{-1}(T_\infty)$ is the unbounded component of the complement of $F^{-1}(F(C))$, and the number of pre-images of a point in T_∞ equals the absolute value of the degree of F.*

The following proposition allows us to count pre-images of regular values (see [8] for pre-images of critical values).

Proposition 1.2. *Let T_l and T_r be tiles to the left and right of a small arc in $F(C)$ with no double points and oriented by sense of folding. Then the points in T_l have two more pre-images then those in T_r.*

For each critical curve γ, we assign an integer $v(\gamma)$ given by the turning of the curve $F(\gamma)$ equipped with the orientation induced by the sense of folding. We recall the definition of the *turning number* $\tau(f)$ of a continuous, locally injective function $f : M \to \mathbb{R}^2$, where M is homeomorphic to S^1 and oriented. Let $g : S^1 \to M$ be an orientation-preserving homeomorphism. Then $\tau(f)$ is the topological degree of the map $\theta \mapsto ((f \circ g)(\theta+\delta) - (f \circ g)(\theta))/|(f \circ g)(\theta + \delta) - (f \circ g)(\theta)|$, where $\theta \in S^1$ and δ is any sufficiently small positive angle so that $(f \circ g)|_{[\theta,\theta+\delta]}$ is injective for all $\theta \in S^1$ (see, e.g., [3]). So, for instance, if $f : S^1 \to \mathbb{R}^2$ is smooth, with $f'(\theta) \neq 0$, for all $\theta \in S^1$, then $\tau(f)$ is the topological degree of $\theta \mapsto f'(\theta)/|f'(\theta)|$. Also, for a continuous locally injective function $f : M \to \mathbb{R}^2$, where M is a disjoint finite union of oriented simple closed curves, define $\tau(f)$ to be the sum of the turning numbers of the restrictions of f to the connected components of M. We now define $v(\gamma)$ as the turning number $\tau(f)$, where f is the restriction of F to the critical curve γ, and γ is oriented by the sense of folding. If $Y = \bigcup_i \gamma_i$, set $v(Y) = \sum_i v(\gamma_i)$. As usual, the *winding* of f around a point $a \in \mathbb{R}^2 - f(M)$ is the topological degree of the map $\theta \mapsto ((f \circ g)(\theta) - a)/|(f \circ g)(\theta) - a|$. Also, set $k^*(\gamma) =$ number of outward cusps in γ and $k_*(\gamma) =$ number of inward cusps in γ.

Proposition 1.3. *Let F be a nice function having k cusp points in C. Then*

$$|\deg(F)| = k - 2v(C) + 1.$$

For a connected component of $\mathbb{R}^2 - C$, with exterior bounding curve γ_0 (if any) and interior bounding curves $\gamma_1, \ldots, \gamma_h$ (again, if any), there holds

$$v(\gamma_0) = 1 + k_*(\gamma_0) + \sum_{i=1}^{h} (k^*(\gamma_i) - v(\gamma_i) - 1).$$

If there is no exterior bounding curve, the left-hand side has to be interpreted as $|\deg(F)|$.

We will refer to these identities as *counting relations*. As an example, consider the function

$$F(x,y) = (-6x^4 - 6x^2y^2 + xy^3 + 6y^4 - x, \frac{25}{24}x^4 + x^3y + x^2y^2 + \frac{1}{6}xy^3 - y^4 - y).$$

The sets C, $F(C)$ and $F^{-1}(F(C))$ are sketched in Figure 1.1. From Proposition 1.3, $|\deg(F)| = 5 - 2 \cdot 2 + 1 = 2$ and the annulus at infinity in the domain is an orientation-reversing double cover of the annulus at infinity in the image. From Propositions 1.1 and 1.2, the number of pre-images of regular points is as indicated in the figure. Let us check the counting relations in this example: for the bounded connected component of $\mathbb{R}^2 - C$, we have $v(\gamma_0) = 2$ and $k_*(\gamma_0) = 1$ (γ_0 being the only critical curve), in agreement with the formula. For the unbounded component, $2 = |\deg(F)| = 1 + (4 - 2 - 1) = 1 + (k^*(\gamma) - v(\gamma) - 1)$.

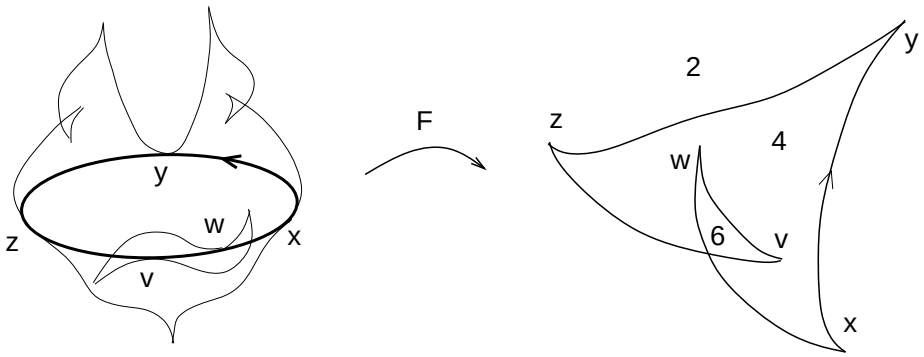


Figure 1.1

On our way to Theorem 1.6, we describe the Blank-Troyer theory ([11, 13]), which addresses the following problem. Let A be a disk with k holes in the plane and $f : \partial A \to \mathbb{R}^2$ be a smooth generic immersion (the precise requirements are described in [7]): when is it possible to extend f to an orientation-preserving immersion $F : A \to \mathbb{R}^2$?

Orient the outer boundary α_0 of A positively and the inner boundaries $\alpha_1, \ldots, \alpha_k$ negatively. A *ray* is a proper embedding $r : [0, +\infty) \to \mathbb{R}^2$ which is transversal to $f(\partial A)$ and never goes through an intersection point of $f(\partial A)$. The images of rays, oriented by the given parametrization, are also called rays. A *system of rays* for f is a finite family of disjoint rays with the following properties:

- the origin of each ray is in some bounded component of $\mathbb{R}^2 - f(\partial A)$,
- each bounded component of $\mathbb{R}^2 - f(\partial A)$ contains the origin of a unique ray.

Intersections between rays and $f(\partial A)$ are labelled *positively* if the curve crosses the ray from right to left and *negatively* otherwise. Each intersection also receives a *height index*: the number of intersections in the same ray which are strictly closer to its origin. The *Blank word* w_i for $f(\alpha_i)$ is constructed by following the orientation of $f(\alpha_i)$ and collecting intersections with the rays, keeping track of rays, signs and height indices: it is defined only up to cyclic permutation.

In Figure 1.2, where $k = 1$, we illustrate a system of rays for which the Blank words are

$$w_0 = a_0^+ b_0^+ c_1^+ d_1^+ e_1^+ f_1^+ b_1^+ c_2^+ d_2^+ e_2^+ f_2^+,$$
$$w_1 = c_0^- d_0^+ e_0^- f_0^-;$$

subscripts indicate height indices.

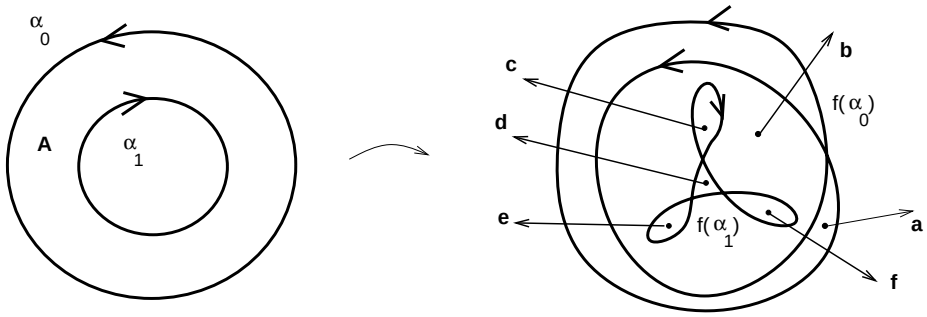


Figure 1.2

Suppose the words w_i and w_j have letters $\mathbf{z}_m^+$ and $\mathbf{z}_n^-$ with $n < m$, so that the two intersections are on the same ray and the one with negative label is closer to the origin of the ray. In order to obtain a *positive concatenation* of w_i and w_j, begin by cyclically permuting w_i (resp. w_j) to make $\mathbf{z}_m^+$ (resp. $\mathbf{z}_n^-$) its last (resp. first) letter, then, juxtapose both words and eliminate the pair $\mathbf{z}_m^+ \mathbf{z}_n^-$. Similarly, a word admits a *positive simplification* if there exists a pair $\mathbf{z}_m^+, \mathbf{z}_n^-$, with $n < m$, such that (after a cyclic permutation if necessary) there are no letters with negative exponent between $\mathbf{z}_m^+$ and $\mathbf{z}_n^-$. In this case, the simplified word is obtained by eliminating the subword $\mathbf{z}_m^+ \dots \mathbf{z}_n^-$ (or $\mathbf{z}_n^- \dots \mathbf{z}_m^+$). Finally, the Blank words $w_0, w_1, \dots, w_k$ admit a *positive grouping* if a sequence of positive concatenations and simplifications leads to a single word with no negative indices.

Back to Figure 1.2, the Blank words admit a positive grouping:

$$\overbrace{c_0^- d_0^+ \; \overbrace{e_0^- (f_0^- \; f_1^+) b_1^+ c_2^+ d_2^+ e_2^+} \; f_2^+ a_0^+ b_0^+ c_1^+} \; d_1^+ e_1^+.$$

Theorem 1.4 (Troyer [13]). *Let A be a k-holed disk, $f : \partial A \to \mathbb{R}^2$ be a generic immersion and consider a system of rays for f. Then there is an extension of f to an orientation-preserving immersion $F : A \to \mathbb{R}^2$ if and only if*

(a) *the turning number of f equals $1 - k$,*

(b) *the Blank words $w_0, w_1, \ldots, w_k$ admit a positive grouping.*

Figure 1.3 should give an idea of how concatenations and simplifications can be used to build immersions: introduce cuts in the domain and image according to the grouping so as to split A into disks in which the immersion is obvious. Conversely, given an immersion, inverse images of rays induce a positive grouping.

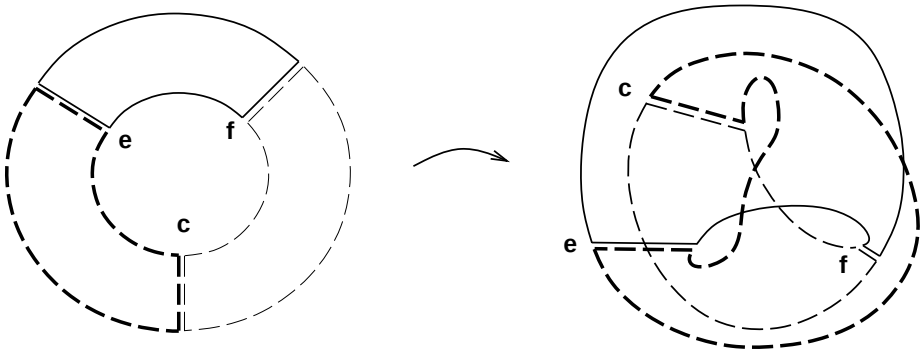


Figure 1.3

Let us now state the analogue of Troyer's theorem for an unbounded region A_∞ whose boundary consists of k smooth simple closed negatively oriented curves $\alpha_1, \ldots, \alpha_k$ (for proofs and more general regions, see [7]). The notion of topological degree naturally extends to a proper continuous function $F : A_\infty \to \mathbb{R}^2$: count, with sign, pre-images of a sufficiently remote regular value. Let $f : \partial A_\infty \to \mathbb{R}^2$ be a generic immersion. In this context the natural question is: given n, can we extend f to a proper immersion $F : A_\infty \to \mathbb{R}^2$ with $\deg(F) = n$?

We need one extra ingredient, the *word at infinity*, which plays the role of the word for the (now nonexistent) exterior boundary. Draw a system of rays as previously discussed. Construct the word at infinity w_∞ by taking the letters corresponding to the rays, running through them counterclockwise as they arrive at infinity, each letter being assigned a positive label and a height index equal to infinity, and finally repeating the pattern $|n|$ times.

Corollary 1.5 [7]. *Let A_∞ be as above, $f : \partial A_\infty \to \mathbb{R}^2$ be a generic immersion and consider a system of rays for f. Then there is an extension of f to an orientation-preserving proper immersion $F : A_\infty \to \mathbb{R}^2$ with $\deg(F) = n$ if and only if*

(a) *the turning number of f equals $1 - n - k$,*

(b) *the words $w_\infty, w_1, \ldots, w_k$ group positively.*

As an example of this result, take A_∞ to be the region outside α_1 in Figure 1.2 with the same f, now restricted to α_1. A system of rays consists of rays c, d, e and f and, for $n = 2$,

$$w_\infty = c_\infty^+ d_\infty^+ e_\infty^+ f_\infty^+ c_\infty^+ d_\infty^+ e_\infty^+ f_\infty^+.$$

The words group essentially as before. In fact, A_∞ and the new immersion F are obtained from the previous example by "pushing α_0 to infinity".

We are now ready to describe the criterion used by the program to decide if a given set $C_\bullet$ of critical curves of the nice function F under study may be its full critical set.

We want to apply Theorem 1.4 or Corollary 1.5 to each connected component R of $\mathbb{R}^2 - C_\bullet$. The only difficulty is the presence of cusps, which we shall handle (following [3]) by applying the above results to sets A, obtained from each R by removing thin tubular neighborhoods of its boundary components.

The topological properties of the image under F of a boundary curve α_i of A are obtained from the nearby critical curve γ_i in the boundary of R by taking into account the sense of folding of F together with the position and type of the cusps. From the normal form of cusps, inward cusps on γ_0 (the outer boundary of R) create small negatively oriented loops in $F(\alpha_0)$; outward cusps, on the other hand, have no such effect. For the other boundary components of R, inner cusps are ineffective and outer cusps again give rise to negatively oriented loops. We call inward cusps on the outer boundary (of R) and outward cusps on inner boundaries *relevant* inside R. In Figure 1.4, for example, the critical curve γ_1 has three outward cusps: if R is the annulus, A and $F(\partial A)$ are as indicated.

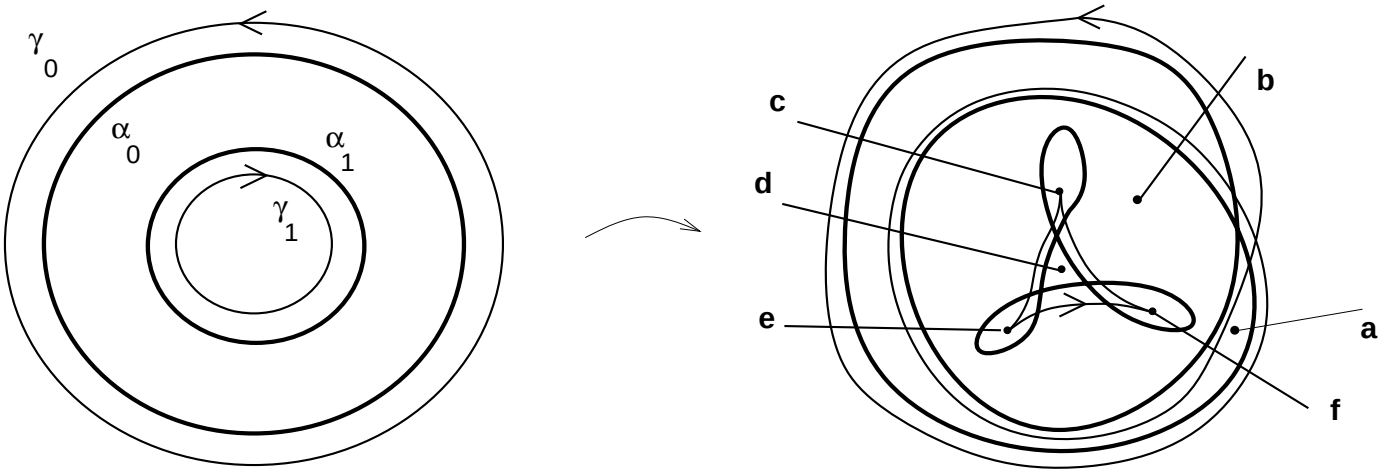


Figure 1.4

In Figure 1.4, the complements of the curves $F(\gamma_1)$ and $F(\alpha_1)$ are rather similar: there is one more connected component associated with each relevant cusp in γ_1. This indicates how to obtain the turning numbers and Blank words for $F|_{\partial A}$ using only $F|_{\partial R}$: there is no need to consider A explicitly. As for turnings, we have $v(\alpha_i) = v(\gamma_i) - k$, where k is the number of relevant cusps (in R) on γ_i. A system of rays for A consists of rays starting from

each connected component of $\mathbb{R}^2 - F(\partial R)$ that do not intersect $F(\partial R)$ at cusps together with rays starting from each relevant cusp. Intersections between rays and $F(\partial A)$ occur near intersections with $F(\partial R)$ and there is one negatively labelled intersection near the origin of each ray starting at a cusp. Height indices are now easily obtained, as are Blank words for the α_i's. In Figure 1.4, we illustrate this construction: Blank words are the same as in Figure 1.2.

Let then $C_\bullet$ be a set of critical curves of F and $R_1, \ldots, R_m, R_\infty$ be the connected components of $\mathbb{R}^2 - C_\bullet$. Each R_ℓ has exterior boundary $\gamma_{[\ell,0]}$ and interior boundaries $\gamma_{[\ell,1]}, \ldots, \gamma_{[\ell,n_\ell]}$ and R_∞ is the unbounded component with (interior) boundaries $\gamma_{[\infty,1]}, \ldots, \gamma_{[\infty,n_\infty]}$. Also, let k_ℓ be the number of relevant cusps inside R_ℓ, i.e., $k_\ell = k_*(\gamma_{[\ell,0]}) + \sum_{j=1}^{n_\ell} k^*(\gamma_{[\ell,j]})$ for $\ell = 1, \ldots, m$ and $k_\infty = \sum_{j=1}^{n_\infty} k^*(\gamma_{[\infty,j]})$. Consider finally for each connected component R_i of $\mathbb{R}^2 - C_\bullet$ a system of rays (for A_i, obtained from R_i by removing a tubular neighborhood of the boundary) and the corresponding Blank words.

An obvious necessary condition for $C = C_\bullet$ is as follows. For each ℓ, $\ell = 1, \ldots, m$, $\det(DF)$ must have constant sign in R_ℓ: in particular, the signs of $\det(DF)$ immediately inside of R_ℓ next to each boundary component $\gamma_{[\ell,j]}$ have to be equal. When this happens, we say signs in R_ℓ are *coherent*. For the unbounded component, signs are coherent if the above signs coincide with that of $\deg(F)$.

Theorem 1.6. *Let F be a nice function, $C_\bullet$, R_ℓ and a system of rays as above. Then $C_\bullet$ is the critical set of some nice function $F_\bullet$ which agrees with F in a neighborhood of $C_\bullet$ and such that $\deg(F_\bullet) = \deg(F)$ if and only if:*

(a) *For $\ell = 1, \ldots, m, \infty$, signs in R_ℓ are coherent.*

(b) $k_\ell - \mathrm{v}(\gamma_{[\ell,0]}) - \sum_{j=1}^{n_\ell} \mathrm{v}(\gamma_{[\ell,j]}) - n_\ell + 1 = 0$ *for $\ell = 1, \ldots, m$, and* $k_\infty - \sum_{j=1}^{n_\infty} \mathrm{v}(\gamma_{[\infty,j]}) - n_\infty + 1 = |\deg(F)|$ (counting relations).

(c) *For each $\ell = 1, \ldots, m$, the Blank words of R_ℓ group positively; similarly, the Blank words of R_∞ and the word at infinity group positively* (word criteria).

Proof. Let us first see that if $F_\bullet$ exists, then the items above hold. Indeed, (a) is trivial. Let $A_\ell \subset R_\ell$ be constructed as above. Notice that $F_\bullet$ is an immersion when restricted to A_ℓ. Conditions (b) and (c) follow from items (a) and (b) of Theorem 1.4 or Corollary 1.5.

Let us now consider the other implication: given that items (a), (b) and (c) hold, we must construct $F_\bullet$. Again, let A_ℓ be as above. Theorem 1.4, possibly after reversing orientation in the domain, tell us that there exists an immersion coinciding with F on ∂A_ℓ. Similarly, apply Corollary 1.5 to the unbounded component to obtain an immersion with topological degree $\deg(F)$. The function obtained from juxtaposing these immersions satisfies all the requested conditions for $F_\bullet$, except for the fact that it may be nonsmooth along the auxiliary curves in ∂A_ℓ. The function can be smoothed out in order to obtain the desired $F_\bullet$ by classical methods. □

Failure of one of the three conditions ascertains the existence of an additional critical curve in the connected component R being treated. If all conditions are satisfied, it is logically possible, from our knowledge of F (i.e., the behavior of F near $C_\bullet$ and at infinity),

that $C_\bullet$ is the full critical set of F. It is not clear, however, that this is actually the case. The problem is already present in the one-dimensional case: how can we be sure, given a function from the line to itself, that we know all its zeros? Implicitly, we assume the knowledge of a priori estimates which guarantee that there are no zeros outside the set being scanned and that, within this set, the function does not oscillate enough to generate additional zeros.

We provide two nontrivial examples in which the set of critical curves is not found completely, despite the fulfillment by $C_\bullet$ of the conditions above. In Figure 1.5, the knowledge of the critical curve γ_0 and of $\deg(F) = 2$ is not enough to decide whether there are additional critical curves. Indeed, if $\tilde{\beta}$ is a simple closed curve surrounding $F(\gamma_1)$ and $F(\gamma_2)$ (the images of the undetected critical curves) its pre-image contains a simple closed curve β surrounding γ_1 and γ_2, as in the figure. If the search for critical curves never probes inside the disk D bounded by β, there will be no reason to expect additional critical curves: clearly, there is a nice function $F_\bullet$ which coincides with F outside D and is a diffeomorphism inside it. On the other hand, we could have found only γ_1, which again gives us no indication of the existence of other critical curves, despite the fact that γ_1 is in many senses different from γ_0: for instance, points surrounded by $F(\gamma_0)$ have four pre-images while points surrounded by $F(\gamma_1)$ have six.

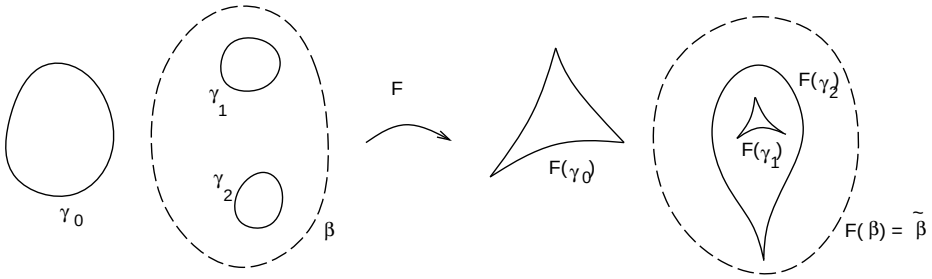


Figure 1.5

There is a characterization analogous to Theorem 1.6 for the critical sets of generic proper Whitney functions whose critical set is unbounded (see [7]). Elsewhere, we intend to consider the extension of this result, and the related numerical analysis, to functions on bounded domains and, by triangulation, on arbitrary surfaces.

2. Computation of the critical set

We describe how the program computes the critical set of a nice function F, given F and its first partial derivatives. The program is not meant to be optimally fast: we are mainly concerned with the reliability of the results and the stability of the algorithms. Many of the intermediate steps perform rather nonstandard numerical tasks in ways which are new to us, and may well be open to substantial improvement. Good performance is

dependent on the choice of additional data kept in order to avoid repetitive searches. For example, bimonotonic arcs, described below, turned out to be very convenient for routines computing winding numbers, intersections of curves, or finding the nearest computed critical point in the ℓ^∞ norm to a given point. The use of memory is also not meant to be optimal: the amount employed is generous but not overwhelming. Relatively complicated executions of the program never used more than 2 Mbytes. We split the program in a series of steps.

Step 0: Study of the behavior at infinity

The program computes the image of a large circle M centered at the origin. This may indicate if the function is not proper or has unbounded critical set, which is taken to be an error: the program expects functions to be nice. If the function is indeed nice, we obtain its degree as being the winding number of $F|_M$ around the origin. The computation of winding numbers is described in step 3. Of course, the appropriate size of M depends on a priori knowledge of F; for instance, it would suffice for M to enclose all the pre-images of 0 under F.

Step 1: A first search for critical curves

The program computes $\det(DF)$, the determinant of the Jacobian of F, at points in a rectangular grid. The grid is specified by adjustable parameters: it is regular near the origin, and sparser far from it. When two consecutive evaluations of $\det(DF)$ have opposite signs, a routine computes one critical point (i.e., a root of $\det(DF)$) in the segment between both grid points. Another routine then obtains a list of points in that critical curve by a continuation method with variable step. Simultaneously, the program checks that the critical curve being computed consists of regular points of $\det(DF)$. This critical curve probably crosses a number of segments of the grid, and those are marked to avoid repetitions in the search for other critical curves. Once the curve is complete, the program goes back to scanning the grid. At the end of this step, the program has found and computed some critical curves, giving rise to a set $C_\bullet$.

Step 2: Marking relevant points in the critical curves

For each known critical curve in the domain, we compute its image and then identify some special points. By comparing neighboring points, we obtain the local extrema for the x- and y-coordinates, both in the list of points in the domain and the image. The curve then splits in a succession of *bimonotonic arcs*: they are maximal segments of the curve which are monotonic in each coordinate. We also obtain minimal enclosing rectangles with sides parallel to the coordinate axes bounding the critical curves and their images.

We also search for candidate neighbors to a cusp, consecutive points p, q and r, by looking for sudden changes of direction between the vectors $F(p) - F(q)$ and $F(q) - F(r)$. Another routine then performs the accurate analysis of that arc of critical curve. Its final outcome is a refined positioning of the cusp (if there is one), together with two neighbors, one on each side, which are closer to the cusp than the typical step size in this region, but which are not so close as to make their images undistinguishable within machine precision.

Computing cusps accurately:

We compute the approximate directions indicated in Figure 2.1. The cusp is then the extremum of the suggested function from the line to itself: we are left with the known problem ([4, 2]) of finding the maximum of a real-valued function without using its derivative. New neighbors to the cusp are computed separately. By evaluating the function at points near a cusp, one assigns a binary flag indicating if the cusp is inward or outward.

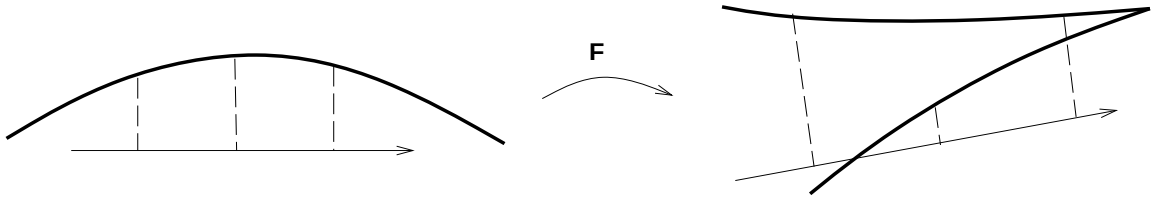


Figure 2.1

We also compute intersection points of curves (actually, this is done only after the computed critical curves have successfully passed the test associated with the counting restrictions to be described in step 4). By hypothesis, the critical curves are disjoint, but their images can meet. In particular, one should also look for points of self-intersection.

Looking for intersections:

Given two images of (possibly equal) critical curves, we first check if their enclosing rectangles meet: it is enough to study the case when this happens. We then consider two bimonotonic arcs, one for each curve: if their enclosing rectangles do not meet, the arcs do not either. In the relevant case, their slopes may have opposite (a) or equal (b) signs, as indicated in Figure 2.2. In case (a), a binary search obtains the only intersection point (if it exists). In case (b), the intersection points are obtained by the algorithm suggested in the picture.

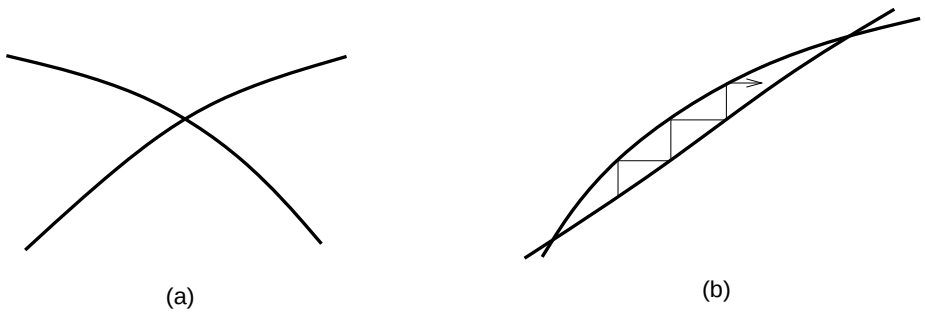


Figure 2.2

At this point, we know rough approximations of intersections, given by the intersections of the polygonals induced by the lists of points. These can be refined, but we do not use Newton's method because that would require knowledge of second derivatives of

F. Instead, from four points a_0, b_0, c_0 and d_0 in the domain approximating the intersection, we obtain (by a method similar to regula falsa) points a_1, b_1, c_1 and d_1 in a similar configuration, giving a better approximation. Iterating this procedure, however, we have to be able to recognize the two nasty situations in Figure 2.3. In the first situation the intersection is not in the interior of the arcs a_0b_0 and c_0d_0 and we must find a new set of four points which is better for topological and not for metrical reasons. In the second situation there is no intersection at all.

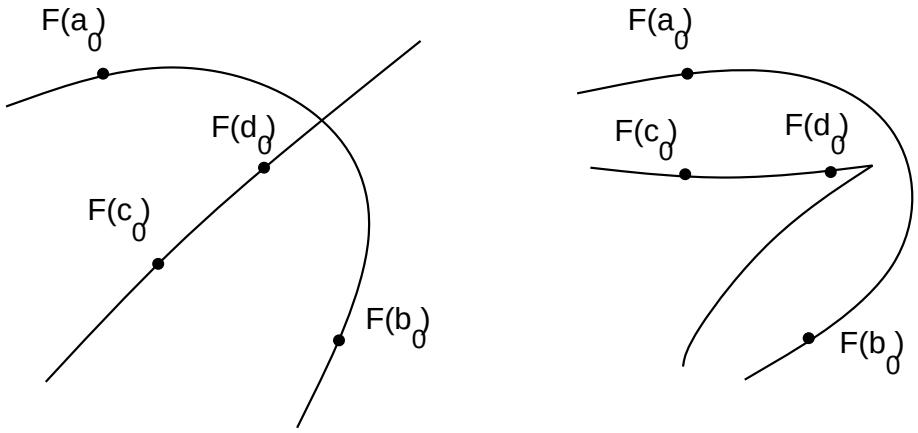


Figure 2.3

Step 3: Study of topological properties of the curves

We recall that the sense of folding of a critical curve γ in the domain is such that $\det(DF)$ is positive immediately to the left of γ. The computation of γ in step 1 proceeds along the sense of folding and we use a binary flag to indicate whether this is the usual positive orientation. This is determined by examining a neighborhood of the point whose image is a maximum of the x-coordinate (such points have been computed in the previous step).

Computing winding numbers:

We explain how to compute the winding number of a curve with respect to the origin of the plane. Often, the endpoints of a bimonotonic arc lie in the same or adjacent quadrants: changes to an adjacent quadrant add or subtract one quarter of a turn from the winding number in an obvious fashion. Changes to the opposite quadrant add or subtract half a turn, and to decide which is the case, we employ a routine that checks if the origin lies above or below the arc.

Computing turning numbers:

Notice that all secant vectors with endpoints in the same bimonotonic arc belong to the same quadrant: we associate this quadrant with the bimonotonic arc. In particular, secant vectors as mentioned in the definition of turning numbers also belong to the quadrant associated with the corresponding bimonotonic arc, provided they are entirely contained inside this arc. Computing the contribution to the turning number corresponding to going over an extreme point at which the curve is smooth is easy: secant vectors for such

consecutive arcs lie in adjacent quadrants and the contribution to the turning number is plus or minus one quarter of a turn. When running past cusps, the argument of the secant vector in the definition of the turning number always changes by (plus) half a turn, when following the sense of folding.

By making use of the routine which computes winding numbers of curves around points, we obtain the inclusion relations among the critical curves in the domain (i.e., which curves lie in the disk bounded by another curve). The same ingredients are used to write a routine which decides if a point is in the region bounded by a set of critical curves.

Step 4: Checking the conditions of Theorem 1.6

During this entire step, there is absolutely no numerical analysis involved, only combinatorial manipulations. We perform sequentially the three kinds of tests associated with the three criteria in Theorem 1.6; we recall that the tests are performed in each component of $\mathcal{R} = \mathbb{R}^2 - C_\bullet$. As soon as one of the tests fails, the program is sent to look for additional critical curves in a specified connected component of $\mathcal{R}$. In the program, each component receives the same label as its exterior bounding curve, the component at infinity receiving by convention the label -1.

The first test (checking if signs are coherent) is performed by comparing the orientation flag attached to each critical curve in the domain. The second test checks the counting relations: ingredients are by now computed. The third test is more complicated: we first compute Blank words, then look for positive groupings.

Let R be a connected component of $\mathcal{R}$; from now on we only consider critical curves in the boundary of R and cusps which are relevant inside R. We first consider the case R bounded. In this step, we call an *arc* the part of an image of a critical curve between two consecutive intersections, equipped with an orientation (both can be chosen), or a full image of a critical curve having no intersections, also with an orientation. Thus, each chunk of image of critical curve between two intersections gives rise to two arcs with opposite orientations. By taking right turns at intersections, we go from an arc to its *successor*. A *cycle* is a closed sequence of successors.

Let $X = \mathbb{R}^2 - F(\partial R)$. Clearly, boundaries of connected components of X are cycles. Also, a cycle is an exterior boundary of a connected component if and only if it is oriented clockwise. We introduce a fictitious cycle at infinity, ordered clockwise, so that all connected components of X have an exterior boundary. It is easy to check inclusions among cycles by looking at windings; this tells us which internal boundaries correspond to a given external boundary.

For each bounded component of X, we will choose one of the arcs on its exterior boundary, the *exit* arc, to be the one through which the Blank rays will leave. Exit arcs are chosen repeating the following procedure until all bounded components of X have been taken care of. For each such component, we check if one of the arcs on its exterior boundary, when seen with reverse orientation, is part of the boundary of either the unbounded component or a bounded component which already has an exit arc; this being the case, we assign to this component one such arc as its exit arc. Exit arcs are

indicated by arrows in Figure 2.4. Thus, starting at any bounded component and leaving through the exit arc, we eventually arrive at the unbounded component. In particular, we have a notion of *depth* of a connected component of X: the unbounded component has depth 0.

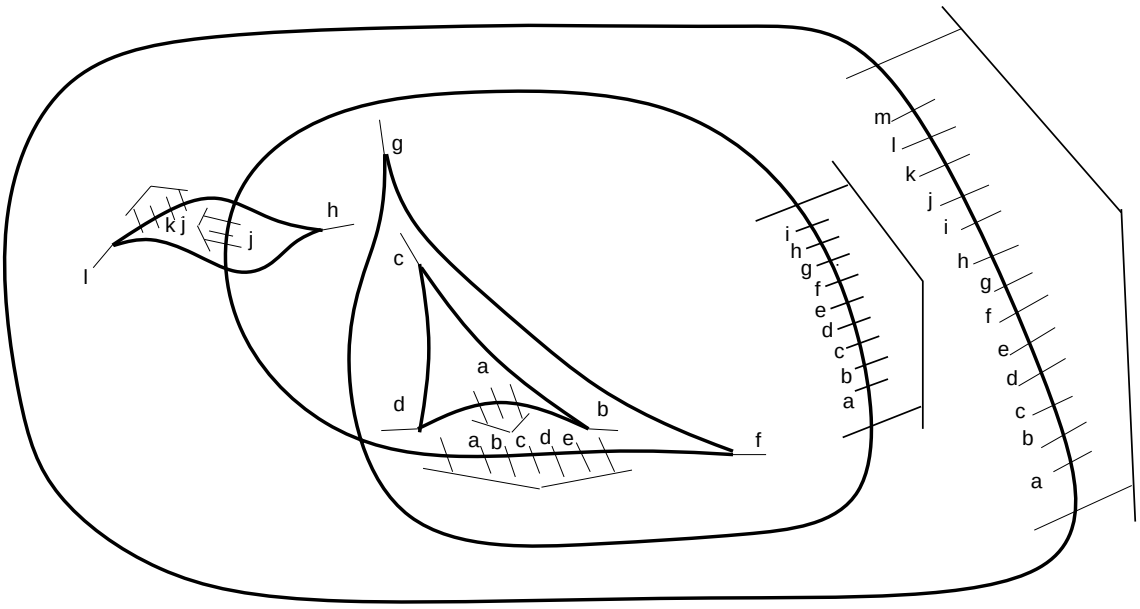


Figure 2.4

The Blank rays can now be easily constructed; actually, the only information we need about the rays is the combinatorial pattern of their intersections with arcs. We start from the deepest connected components of X and proceed to shallower components: in a typical inductive step, there are incoming rays constructed at previous steps, exiting from deeper components to a connected component of X being considered. At this point, we have a list of (incomplete) rays and the sequenced list of their intersections with previous exit arcs, together with height indices. To each group of rays coming from a given boundary component we add a ray for each (relevant) cusp on this cycle. Inside each arc of this cycle, the order of rays in a group is determined by previous steps, and, more globally, by the order of arcs on the cycle. If the cycle is the external boundary, we must pay attention to the position of the exit arc. We force all rays to go out through the exit arc, preserving the order of the rays in each group and assigning an arbitrary order to the groups. We add a last ray for the connected component itself. This process gives us a full description of the combinatorial pattern mentioned above for a system of rays. The Blank word for each curve is now obtained simply by following the curve and keeping track of intersections of rays with its arcs, including the formal intersections at cusps. We already noticed that the sign at cusps is always negative; other signs are obtained by comparing the orientation of the exit arc with the sense of folding of the curve.

Finally, search for a positive grouping — while potentially time consuming, this is very simple in our usual examples. A slight simplification is obtained by ignoring rays

with no negative labels. This takes care of bounded connected components R. The word at infinity, necessary for the unbounded component R_∞, is easy to construct.

Step 5: Looking for additional critical curves

It may happen that a connected component R in $\mathcal{R}$ failed some of the previous tests. We search for a critical point there as follows. The program looks for two points p and q in R at which $\det(DF)$ has different signs and such that the segment pq is entirely contained in R (equivalently, it intersects no known critical curve). This is necessary to guarantee that we actually find a new critical curve. Notice that, in some nasty cases, the procedure can be time consuming.

Suppose R is bounded. Boundary curves of R for which the sign of $\det(DF)$ immediately to the interior of R is the same as that for the exterior boundary are called *good* and the others, *bad*. Violation of sign coherence corresponds to the existence of a bad curve. In this case, let p' be the point on the bad boundary components with the lowest y-coordinate. Consider the points on the good boundary components with y-coordinate smaller or equal than that of p'. Among these, take q' to be the closest to p' in the ℓ^∞ norm (the chosen norm substantially simplifies the search for the closest point: it is either the intersection of the arc with a straight line passing through p with slope ± 1 or one of the extremes of the arc). The points p and q are obtained by pulling p' and q' slightly closer to each other along the segment $p'q'$. The case where R is unbounded is similar and left to the reader.

From now on, we assume that signs in R are coherent. Let p be a random point in R. We say that a point p is *easy* if $\det(DF)(p)$ has the same sign as $\det(DF)$ for points of R near the exterior bounding curve γ (or, if this is the unbounded component, the same sign as the topological degree of F). If p is not easy, we can pick q simply by finding that point in the boundary of R which lies closest to p in the ℓ^∞ norm and then pulling it slightly closer to p. If p is an easy point, we perform a modified Newton method (described below) on the function $\det(DF)$ to find a new point which is hopefully not easy; this process may be iterated a limited number of times before the program either succeeds or gives up the search starting from p and tries a new random point.

The modified Newton method:

Given an easy point p, we find an approximation u to the gradient of $\det(DF)$ on p by computing finite differences. In the line through p with the direction of u, we solve for p' the equation for the linearization of $\det(DF)$:

$$\det(DF)(p) + u \cdot (p' - p) = -\alpha \det(DF)(p).$$

The parameter α is a number between 0 and 1. We then check if p' is still in R.

Whenever a new critical curve is found, the program repeats the necessary parts of steps 2 to 5, until all conditions are satisfied. At this point, the program considers the computation of the critical sets to be complete.

The program accepts a number of parameters, which have been adjusted rather empirically. Again, we cannot expect our program (nor any algorithm relying strictly on function evaluation) to deal with an arbitrary nice function. Still, our experience indicates that the program behaves quite well in many nasty examples (see §4) and appropriately halts if the function is not nice. The robustness of the program is largely due to our (prudent) variable-step strategy when computing critical curves.

3. Inversion of the function

At the beginning of the inversion procedure, we have at our disposal all the critical curves γ_i, i.e., for each one, a list of points together with their images, including cusps and intersection points (computed close to machine precision) and a collection of flags indicating the following properties of each point:

i. if a point is a cusp, and if the cusp is inward or outward,

ii. if the point is a local extremum of the x- or y-coordinate in the lists of γ_i or $F(\gamma_i)$ (there are four flags of this kind),

iii. if a point is an intersection point.

We are now ready to begin the computation of the solutions of the equation

$$F(x) = a, \qquad x, a \in \mathbb{R}^2. \tag{1}$$

A frequent operation will be the *inversion* along the segment ab with *initial condition* p, a known pre-image of a. We use an extended Newton's method, as described in detail in [1]. There are two ways in which inversion may fail. The segment ab could intersect the image of the critical set at a point c, in which case we choose to interrupt the inversion procedure, returning a value of c. Else, it is of course possible that a numerical error occurs, and the inversion routine then sends back a warning. We actually expect the first kind of 'failure' to occur several times during an execution of the program.

Again, we break the description of the program into steps.

Step 0: Computing all the pre-images of some simple points

Take a point u in the domain just away from the critical set C and such that $F(u)$ is also away from $F(C)$ (this is easily accomplished from information collected by flags of type (ii)). Join its image $F(u)$ by a horizontal or vertical segment I to a nearby point a in one of the diagonals $y = x$ or $y = -x$. Define the points b, c and d in the image to be consecutive right-angle turns of a around the origin. Next, attempt to invert along segment I, with initial condition u. Assuming success, we now have the first pre-image p of the point a. Use p now as an initial condition for inversion along the sides ab, bc, cd and da of the square $abcd$. Again assuming success, we now have another pre-image of the point a. Repeat the cycle of inversions along the sides of the square until we obtain the total number of pre-images of a: their number is $|\deg(F)|$. This procedure is justified by the behavior of F at infinity. If any of the inversion procedures fail, we start again with u

further from the origin (say, twice the distance); again, from the behavior of F at infinity, this method must eventually work. Summing up, we now have four points a, b, c and d along the diagonals all of whose pre-images are known. Algebraically, we solved system $(*)$ for the right hand side equal to each of these four points. Points all of whose pre-images are known will be called *solved* points. Some of these points will be permanently kept for eventual use in a *bank* of solved points.

Step 1: Computing the pre-images of an arbitrary point

Let e be a point in the range. Join e to one of the solved points a in the bank by an L-shaped path, i.e., a path consisting of two segments, one horizontal and one vertical (the order is not important) going from a to e. There are many such paths, and we choose one by favoring the properties below.

1. Both segments should stay away from images of cusps (which are points of potential difficulty for the inversion procedures) and of local extrema of x- and y-coordinates of $F(C)$ (this is requested so that intersections with $F(C)$ are transversal).
2. Both segments should have few intersections with $F(C)$.
3. Segments should be as short as possible.

Once a path is chosen, we look for all intersections of critical curves with this path, and classify them according to orientation: as the path crosses the image of the critical curve, the number of pre-images either increases or decreases by two, as we have seen in Proposition 1.2. We now perform an inversion procedure starting from the solved point a. We do this for each pre-image of a until we either obtain a pre-image of e, or the pre-image of the path approaches a critical curve in the domain, thus leading to an inversion failure of the first kind. We also have to do inversion starting from those intersections c of the path with $F(C)$ where two pre-images appear. We easily compute the point r in C which is sent to c. In order to generate the two pre-images near r, compute the kernel K of $DF(r)$, and choose two points s_1 and s_2 near r in the line $r + K$, one on each side of the critical curve. The images $d_1 = F(s_1)$ and $d_2 = F(s_2)$ most certainly do not belong to the segment being inverted. We then connect d_1 and d_2 to the path by small straight line segments I_1 and I_2, invert them with the obvious initial conditions, and continue inversion along the original path. Finally, we check that failures happened at the right places and in the right amount. Unfortunately, from the information we are keeping we do not know a priori which are the two pre-images doomed to vanish.

If this process as a whole is successful, i.e., if the only failures of the inversion process correspond to the vanishing of pre-images, it gives us all the pre-images of e. A genuine difficulty, however, is that there are cases in which no L-shaped path from the point e to be inverted to points in the bank is acceptable in the sense of request (1) above. One possibility would be to search for more complicated paths joining e to some point in the bank, steering away from dangerous regions of inversion. Another possibility is to wait for the bank of solved points to be sufficiently rich so as to contain a point which can be joined to e by an acceptable L-shaped path. This leads us to the next step.

Step 2. Collecting solved points

It is convenient to keep a bank, consisting of certain points together with their full set of pre-images, initialized with a, b, c and d, as in Step 0. Occasionally, we will add to this bank a new solved point, inverted either by request or for being the corner of the employed L-shaped path. This will permit improvements in the choice of subsequent L-shaped paths, thus making more points accessible to the inversion procedure. There are many reasonable criteria for insertion of a new solved point in the bank but we do not know which is best.

The computation of the flower is a matter of inverting each point in the lists describing $F(C)$. A point is very close to its neighbors in the same curve, so we usually have good initial conditions for its inversion but things do not work so nicely near a cusp or intersection. We therefore proceed arc by arc, where an arc is that portion of one of the curves in $F(C)$ between two such troublesome points. The situation for a typical point inside an arc is very simple to handle numerically: all pre-images but one are far from the critical set C and the remaining one, which is in C, is already known explicitly. We can therefore use our previously discussed inversion procedures to compute all the unknown pre-images of one point about midway in each arc and then invert the rest of the arc by standard continuation methods. Of course, there could be difficulties when the regular pre-images get very close to C: a typical example is a small swallowtail, i.e., a pair of very close cusps, one inward and one outward, on the same critical curve (cusps v and w in Figure 1.1 form a swallowtail).

4. Examples

In this section, we describe several executions of the program. The experiments were performed on a Sparc Station SLC (diskless, 8 Mbytes). We begin by considering the function

$$F(x,y) = (x^5 - 10x^3y^2 + 5xy^4 + 6x^2 + 6xy + x, 5x^4y - 10x^2y^3 + y^5 - y).$$

After running through a rather exhaustive grid of $61 \times 61 = 3721$ points, the program found three critical curves which were traced with 57, 61 and 157 points. These curves and their images are shown in Figure 4.1. The program checked sign coherence and counting relations and proceeded to compute the Blank words: in this example, the regions inside critical curves trivially satify the word criterion. For the unbounded regular region, it obtained the Blank words $a_0^- b_0^- c_0^-$ for curve 0, $d_0^- e_0^- f_0^-$ for curve 1, $b_1^+ c_1^+ d_1^+ e_1^+ f_1^+ g_0^- h_0^- i_0^- j_0^-$ for curve 2 and the word at infinity $b^+c^+d^+e^+f^+g^+h^+i^+j^+a^+b^+c^+d^+e^+f^+g^+h^+i^+j^+a^+b^+c^+d^+$-$e^+f^+g^+h^+i^+j^+a^+b^+c^+d^+e^+f^+g^+h^+i^+j^+a^+b^+c^+d^+e^+f^+g^+h^+i^+j^+a^+$ (height indices for the word at infinity are infinite), which were checked to admit a positive grouping. At this point, the critical set was considered to be completely known. This part of the program took 4.5 seconds: 1.5 seconds were taken by running over the grid, 1.4 seconds were spent in the tracing of the critical curves (in a detail finer than necessary), the precise computation

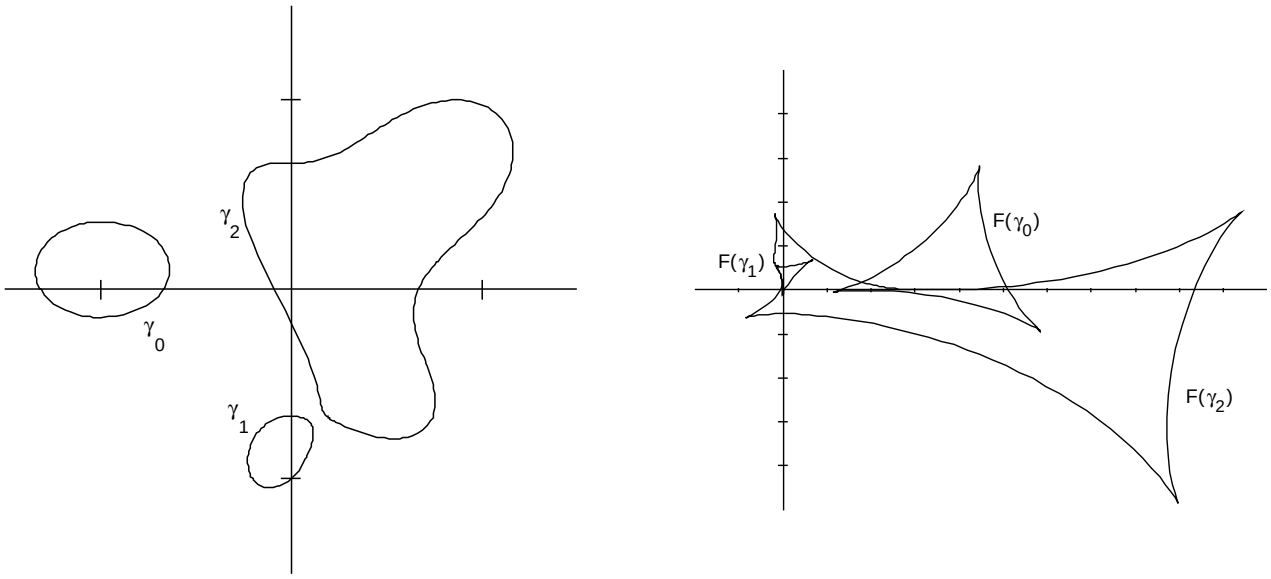
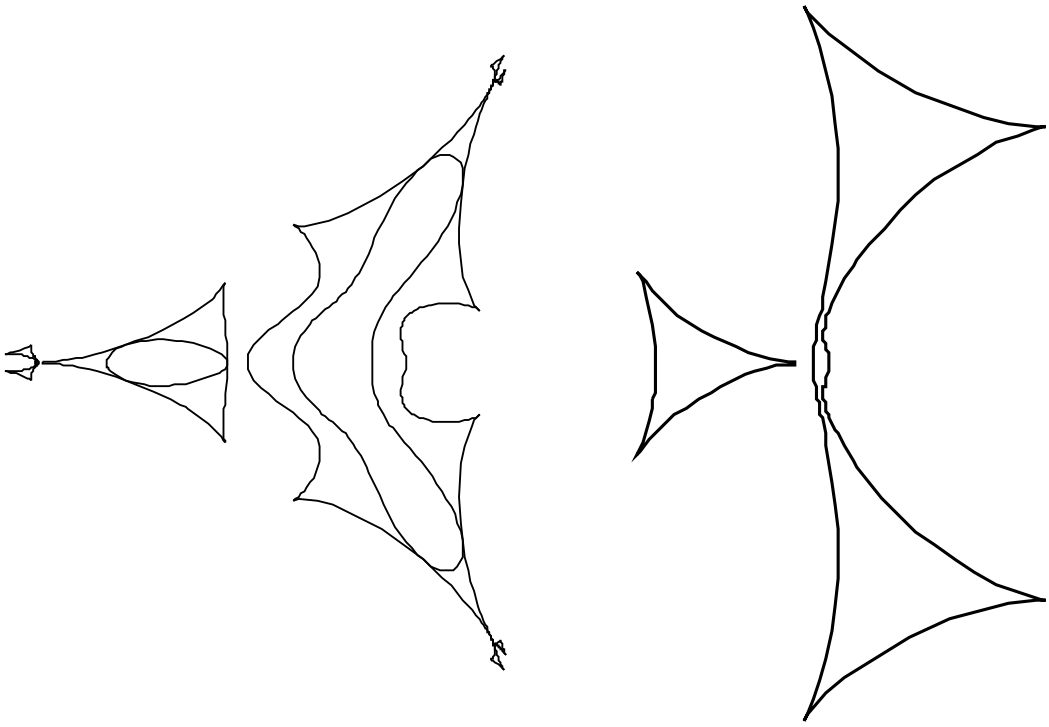


Figure 4.1

of the intersection of the images of the critical curves took 0.2 seconds and the word test took further 0.2 seconds.

We then undertook two experiments to evaluate the performance of the inversion routines. We computed, with our algorithm, the pre-images of 199 uniformly distributed random points in the square $[-1,1]\times[-1,1]$. Notice that this square contains a nontrivial portion of the image of the critical set: there were points with 5, 7 and 9 pre-images. This took the program 14 seconds (plus 2 seconds for input and output). The average number of pre-images per point turned out to be 5.82. We then took other 199 uniformly distributed random points in the square $[-6,-4]\times[-1,1]$, which is quite removed from the image of the critical set. The program then computed all five pre-images of each point, from the previously known pre-images of the point $(-5,0)$ by a simple standard continuation method, which usually turned out to be equivalent to (a nonextended) Newton's method. This part of the program took 2.5 seconds for computations (and 1.6 seconds for input and output). We find the 6:1 ratio of required times satisfactory for the following reasons. The prototype being tested is far from being optimal and the task it performed is substantially harder: a standard continuation method would not be able to ensure that the totality of pre-images had been found. Besides, the segments appearing in the continuations performed in the first part of the test were much closer to the critical set, forcing the predictor-corrector method to be more careful and therefore slower even when the segments did not cross the image of the critical set. Clearly, a larger number of pre-images is responsible for still additional time in the first part of the test. Notice that the program inverted successfully all 199 random points, despite of their probable closeness to the image of the critical set. The total time, including computation of the critical set and the inversion of the 2×199 points was 29 seconds, of which 17 seconds were spent in the nearly 35,000 evaluations of the function and its Jacobian.



(domain) **(image)**

Figure 4.2

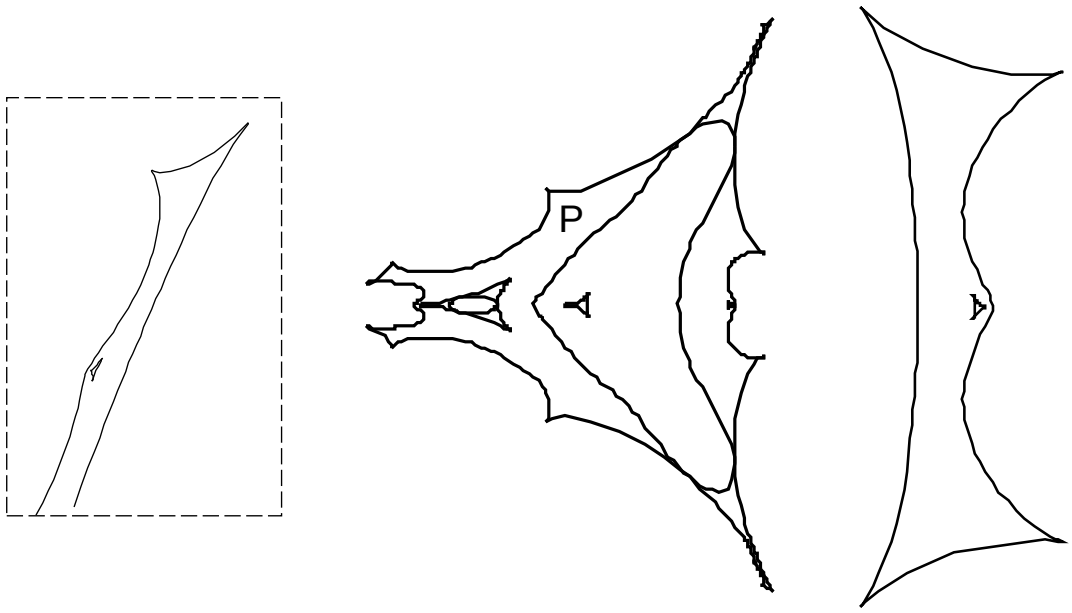
As another experiment, we set a modest square grid of $9 \times 9 = 81$ points for the initial search of critical curves of the same function F defined above. The program started by correctly computing the global degree of the function to be equal to 5. Then, by running over the grid, it found γ_2; four cusps were found on this curve and the turning number of its image was computed to be 1. By checking the counting restrictions, the program correctly concluded that there were critical curves missing in the unbounded component of the complement of γ_2. By the random search method, γ_0 was found and analyzed: curves were still missing. Again, random search found γ_1, and all conditions were shown to be then satisfied.

We now show two examples of use of the program to obtain global understanding of a function by means of the computation of its flower $\mathcal{F} = F^{-1}(F(C))$. Consider the family of functions

$$F_t(x, y) = (t(x^4 - 6x^2y^2 + y^4) + x^3 + xy^2 - 2x, t(4x^3y - 4xy^3) - x^2y - y^3 - 2y).$$

The flowers shown in Figures 4.2 and 4.3 correspond to the values $t = 0.15$ and $t = 0.3$ respectively. Notice the tiny pre-images of the triangles in the image and the presence of a double covering in the restriction of the second function to the region P.

As another experiment, the program inverted some points very near the image of the critical set of a function. More precisely, consider the function $F(x, y) = (x^2 - y^2 + x, 2xy - y)$, with critical set given by the circle centered at the origin with radius $1/2$.

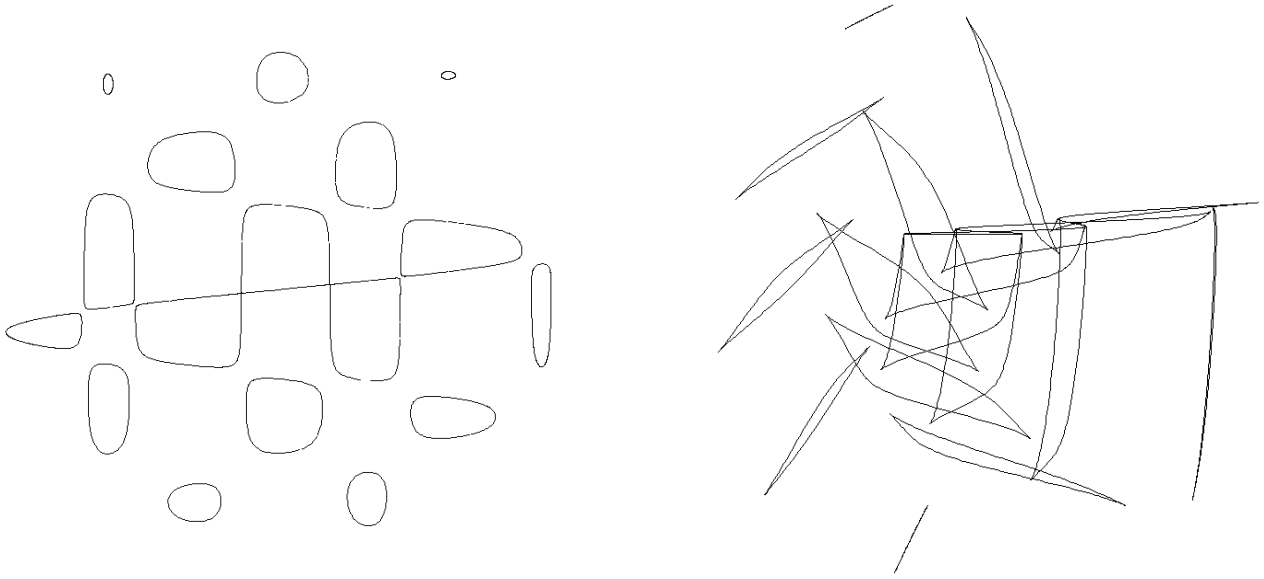


(detail) **(domain)** **(image)**

Figure 4.3

The intersection of the image of the critical set with the positive y-semiaxis is given by $(0, \frac{(3-\sqrt{3})\sqrt[4]{12}}{8}) \approx (0, 0.294989919892746)$, which is the image of a fold point. For points $A = (0, 0.2949899198927)$ and $B = (0, 0.2949899198929)$, the inversion led to correct answers: the first point has four pre-images, $a_1 = (0.643433123248, 1.028318679856)$, $a_2 = (-1.009458527033, -0.097713820754)$, $a_3 = (0.183018329423, -0.465310690167)$ and $a_4 = (0.183007074333, -0.465294168935)$, and the second point has only two-pre-images, which agree with the points a_1 and a_2 to all twelve displayed decimal places. Moreover, applying F to the points a_1 and a_2, we obtained the points A and B with an error of roughly 10^{-13}: at this scale, A and B are undistinguishable. In other words, errors were probably due to truncation only. Moreover, a_3 and a_4 stand on opposite sides of the critical curve: their distance to the critical set is of the order of the square root of the distance of the points A to the image of the critical set, which is the expected behavior of a function close to a fold point. When computing F at a_3 and a_4, however, we missed the point A by an error of the order of 10^{-11}, indicating that full machine accuracy was not achieved but relatively good accuracy was in fact obtained. We then inverted some points very close to $(0.75, 0)$, which is the image of the cusp $(0.5, 0)$. The inversion worked well for the points $(0.75, 10^{-8})$, $(0.75 + 10^{-11}, 0)$ and $(0.75 - 10^{-11}, 0)$.

Our final example is the study of the function $F(x, y) = (x^2 - y^2 + 20 \sin x, 2xy + 20 \cos y)$. The critical set of $(x, y) \mapsto (\sin x, \cos y)$ is a grid of lines of the form $x = k\pi + \frac{\pi}{2}$, $y = k\pi$, where k is an integer. Due to the quadratic term, F behaves at infinity like $z \mapsto z^2$, in complex notation. In particular, the critical set of F is bounded and the topological degree of the function is equal to two. Both C and $F(C)$ are given in Figure 4.4: notice the large number of critical curves in the domain (seventeen) and the intricate pattern of intersections in the image. In the computations for this example, we first estimated a



(domain) **(image)**

Figure 4.4

bound for the critical set and made use of a very fine initial grid. Indeed, only by doing so could we hope to find the two small critical curves at the upper right and left corners in the picture of the domain: they would be undetectable by all tests (recall the examples described in Figure 1.5): this is yet another situation where a priori estimates are needed. The combinatorics related to the grouping of words is overwhelming, at least for this algorithm, due to the many nontrivial Blank words. In this execution, we inhibited the word criterion. Still, the program had no difficulty in inverting the points $(-32, 32)$, $(8, 32)$, $(-8, 16)$, $(8, 16)$ and $(-2, 10)$, which have respectively 2, 2, 8, 10 and 10 pre-images. The last three points were taken from a region in the image particularly crowded with images of critical curves. This function seems to be a hard task for any algorithm to compute (or even count) pre-images: indeed, how is it going to infer that points inside the tiny lip at the upper left corner of the image have four pre-images without spending substantial time scanning the domain for critical curves?

Acknowledgements. The authors thank Antônio Castelo and Hélio Cortes Lopes for help in dealing with computers. We are grateful to the Departamento de Informática, PUC-Rio, which kindly allowed us to use their equipment. This work is supported by CNPq and MCT, Brazil.

Bibliography

1. Allgower, E. L., Georg, K., *Numerical continuation methods: an introduction*, Springer-Verlag, New York, 1991.
2. Fletcher, R., *Practical methods of optimization*, John Wiley and Sons, New York, 1980.

3. Francis, G. K., Troyer, S. F., *Excellent maps with given folds and cusps*, Houston Jr. of Math., vol.3, No. 2 (1977), 165-192.
4. Gill, P. E., Murray, W. and Wright, M. H., *Practical optimization*, Academic Press, New York, 1981.
5. Keller, H. B., *Global homotopies and Newton methods*, in *Recent advances in numerical analysis*, eds. C. de Boor and G. H. Golub, Academic Press, New York, 1979, 73-94.
6. Kubíček, M. and Marek, I., *Computational methods in bifurcation theory and dissipative structures*, Springer-Verlag, New York, 1983.
7. Malta, I., Saldanha, N. C., Tomei, C., *Critical sets of proper Whitney functions in the plane*, preprint.
8. Malta, I., Tomei, C., *Singularities of vector fields arising from one dimensional Riemann problems*, J. Diff. Eq., 93 (1991).
9. Milnor, J. W., *Topology from the differentiable viewpoint*, University Press of Virginia, 1969.
10. Morgan, A. P., *Solving polynomial systems using continuation for engineering and scientific problems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1987.
11. Poénaru, V., *Extending immersions of the circle (d'après Samuel Blank)*, Exposé 342, Séminaire Bourbaki 1967-68, Benjamin, NY, 1969.
12. Quine,J.R., *A global theorem for singularities of maps between oriented 2-manifolds*, Trans. AMS, vol.236 (1978), 307-314.
13. Troyer, S.F., *Extending a boundary immersion to the disk with n holes*, Ph. D. Dissertation, Northeastern U., Boston, Mass., 1973.
14. Whitney, H.: *On singularities of mappings of Euclidean spaces, I: mappings of the plane into the plane*, Ann. of Math. 62 (1955), 374-410.

Iaci Malta, PUC-Rio
Nicolau C. Saldanha, IMPA and PUC-Rio
Carlos Tomei, IMPA and PUC-Rio

Departamento de Matemática, PUC-Rio
Rua Marquês de São Vicente 225, Rio de Janeiro 22453-900, Brasil

IMPA, Estr. Dona Castorina 110, Rio de Janeiro 22460-320, Brasil

malta@mat.puc-rio.br
nicolau@impa.br, http://www.impa.br/∼nicolau/
tomei@impa.br