SIAM J. Sci. Comput.
Vol. 19, No. 6, pp. 1783–1803, November 1998

NUMERICAL CONFORMAL MAPPING USING CROSS-RATIOS AND DELAUNAY TRIANGULATION*

TOBIN A. DRISCOLL† AND STEPHEN A. VAVASIS‡

Abstract. We propose a new algorithm for computing the Riemann mapping of the unit disk to a polygon, also known as the Schwarz–Christoffel transformation. The new algorithm, CRDT (for cross-ratios of the Delaunay triangulation), is based on cross-ratios of the prevertices, and also on cross-ratios of quadrilaterals in a Delaunay triangulation of the polygon.

The CRDT algorithm produces an accurate representation of the Riemann mapping even in the presence of arbitrary long, thin regions in the polygon, unlike any previous conformal mapping algorithm. We believe that CRDT solves all difficulties with crowding and global convergence, although these facts depend on conjectures that we have so far not been able to prove. We demonstrate convergence with computational experiments.

The Riemann mapping has applications in two-dimensional potential theory and mesh generation. We demonstrate CRDT on problems in long, thin regions in which no other known algorithm can perform comparably.

Key words. numerical conformal mapping, Schwarz–Christoffel mapping, cross-ratios, Delaunay triangulation

AMS subject classifications. 30C30, 65E05

PII. S1064827596298580

1. Introduction.

Let P be an open, simply connected, proper subset of the complex plane $\mathbf{C}$. The celebrated Riemann mapping theorem [8] states that there is an analytic function f with a nowhere-vanishing derivative such that $f(D) = P$, where D denotes the open unit disk, and such that f is bijective on D. Furthermore, let z_0 be an arbitrarily specified point in P and let α be an arbitrary angle in $[0, 2\pi)$. Then f can be chosen so that $f(0) = z_0$ and $\arg f'(0) = \alpha$. With such a specification, f is uniquely determined. The point z_0 is called the *conformal center* of f.

This mapping f can be used to solve problems in potential theory posed on P. The classical example is Laplace's equation $\triangle u = 0$, which is invariant under conformal transplantation. A second application is finite-difference mesh generation. Because conformal mapping preserves angles, an orthogonal grid on the disk is mapped by f to a grid on P whose grid lines also meet orthogonally, which simplifies the task of discretizing a differential operator.

In the case that P is a simple polygon, the Riemann mapping can be written down almost in closed form. Let P be a finite polygon whose boundary is piecewise linear, with no interior angles equal to 0 (no cusps). Let the vertices of P in counterclockwise order be denoted $z_1, \ldots, z_n$. Let the interior angles at $z_1, \ldots, z_n$ be $\alpha_1, \ldots, \alpha_n$, and

*Received by the editors January 29, 1996; accepted for publication (in revised form) November 29, 1996; published electronically July 27, 1998.
http://www.siam.org/journals/sisc/19-6/29858.html

†Department of Applied Mathematics, University of Colorado, Boulder, CO 80309 (tad@colorado.edu). This work was funded in part by DOE grant DE-FG02-94ER25199 to L. N. Trefethen and by an NSF Postdoctoral Research Fellowship.

‡Department of Computer Science, Cornell University, Ithaca, NY 14853 (vavasis@cs.cornell.edu). This author's work was supported in part by an NSF Presidential Young Investigator grant with matching funds from Xerox, AT&T, Sun, and Tektronix. This work was partially supported by Xerox Corporation, during the author's visit to the Xerox Palo Alto Research Center. This work was partially supported by the Research Institute for Advanced Computer Science (RIACS) during the author's visit to RIACS, sponsored by NASA under contract NAS 2-13721. This work was partially supported by Argonne National Laboratory while this author was visiting.

let $\beta_j = \alpha_j/\pi - 1$ for each j, so that $\beta_j \in (-1, 1]$ for each j. (If β_j=1, z_j is the tip of a slit, and the sides adjacent to z_j coincide at least partially.) Then any conformal mapping $f : D \to P$ has the form [8]

$$f(w) = A + B \int_0^w \prod_{j=1}^{n} \left(1 - \frac{\omega}{w_j}\right)^{\beta_j} d\omega, \tag{1}$$

where A, B are complex constants ($B \neq 0$) and $w_1, \ldots, w_n$ are points in counterclockwise order on the boundary of the unit disk. The points $w_1, \ldots, w_n$ are called *prevertices*; they map to the points $z_1, \ldots, z_n$. Formula (1) is known as the *Schwarz–Christoffel (S–C) formula*.

The S–C formula is not quite in closed form, because there is no explicit expression for the $n+4$ real parameters $A, B, \theta_1, \ldots, \theta_n$, where $\theta_i = \arg w_i$. The affine constants A and B can be determined after the prevertices are known, and the three degrees of freedom may be fixed, so that $n - 3$ unknowns remain. In practice, these parameters are determined by solving a system of nonlinear equations derived from geometric constraints. Any particular specification of the unknown parameters will yield some polygon whose side lengths and orientation can be measured and compared to the desired image polygon to yield $n - 3$ conditions [19].

Actual software packages for S–C mapping like SCPACK [18] and its cousin, the SC Toolbox for MATLAB [7], solve such nonlinear systems numerically. But two difficulties limit the generality of polygons these packages can handle.

1. The system of nonlinear equations does not have any special structure that lends itself to easy solution. In fact, the system can have local minima that can trap nonlinear solvers and prevent convergence entirely [9].

2. More important, SCPACK and the SC Toolbox cannot generally handle *crowding*, a phenomenon of conformal mapping that occurs whenever the domain has any long, narrow channel [14]. Such a channel will skew the prevertex positions to a degree exponential in the aspect ratio of the narrow region (see Fig. 2). For instance, if the aspect ratio is 20 to 1, then at least 14 significant digits are lost when computing a difference between certain θ_j's.

One partial solution to the problem of crowding in the SC Toolbox is its provision for mapping elongated domains to rectangles. The elongation in the domain can be matched by a similar elongation in image rectangle, alleviating the crowding problem [9, 12]. While this technique can be generalized to certain classes of multiply elongated polygons [11], the technique becomes much more delicate, and a new fundamental domain is needed for each number of elongated branches.

We propose a new algorithm that remedies both of the deficiencies described above. There are two principal innovations in the new algorithm.

1. Our algorithm uses as primitive variables certain *cross-ratios* of the w_j's (see section 4). Because cross-ratios are invariant under fractional linear transformations, we can compute many different conformally equivalent embeddings of the w_j's, each of which yields the same polygon. In particular, we choose embeddings for which the resulting maps are accurate locally within different regions of the polygon. Thus, crowding is no longer a problem.

2. Our system of equations enforces constraints on certain absolute cross-ratios in the image polygon (rather than enforcing conditions about side lengths and orientations as above). The resulting nonlinear system appears to have a monotonicity property that makes it much easier to solve than the other formulations.

To keep the cross-ratios near $O(1)$, some edges of the polygon will first be split; i.e., new vertices are introduced whose angles are π. In order to define both sets of cross-ratios, the CRDT algorithm computes a Delaunay triangulation which defines $n - 3$ overlapping quadrilaterals. CRDT stands for "cross-ratios of the Delaunay triangulation."

The remainder of the paper is organized as follows. In section 2 we describe the basic properties of the constrained Delaunay triangulation. In section 3 we present the splitting step of our algorithm. In section 4 we define cross-ratios and establish some of their properties. In particular, we show that $n-3$ cross-ratios uniquely determine the image polygon under (1) up to similarity transform. In section 5 we present the whole algorithm in detail. In section 6 we explain how CRDT circumvents crowding. In section 7 we discuss solvers for the nonlinear system and report on experiments with various polygonal domains. In section 8 we describe how to use CRDT to compute the disk map. In section 9 we describe how to use CRDT in two applications. We know of no other algorithm that could duplicate the results of section 9.

2. The Delaunay triangulation.

Let P be a simple polygon. A *triangulation* of P is a division of P into nondegenerate triangles whose vertices are vertices of P. Triangles that intersect must do so at either a vertex or an entire edge.

It is well known that any n-vertex simple polygon P has a triangulation consisting of $n-2$ triangles. It is also known that any triangulation of P has exactly $n-3$ distinct *diagonals*, triangle edges that are not also polygon edges. Each triangulation has a *dual* graph, which has one node for each triangle and an edge between two nodes if their corresponding triangles have a common diagonal. It is well known that the dual graph of a triangulation of a simple polygon is in fact a tree.

Every P has at least one *constrained Delaunay triangulation* [2], or just *Delaunay triangulation*, which has the following (defining) property. If d is a diagonal, let $Q(d)$ be the quadrilateral associated with d, that is, the union of the two triangles on either side of d. Then the sum of the two opposite interior angles of $Q(d)$ that are split by d is at least π. A Delaunay triangulation of P can easily be computed in $O(n^2)$ steps.

3. Splitting edges.

The first step of our algorithm is to split some edges of the polygon. Splitting an edge means replacing it by several smaller edges whose union is equal to the original edge. These new edges are joined by vertices whose angles are π. Notice that this operation does not affect the S–C formula (1); a vertex whose interior angle is π has its β exponent equal to 0 in that formula.

The purpose of splitting is to make sure each individual quadrilateral in the Delaunay triangulation is well conditioned. By "well conditioned" we mean that the prevertices of the quadrilateral are not too crowded in some valid arrangement of the S–C prevertices. In particular, we want to avoid quadrilaterals that are long and narrow with the long edges equal to polygon edges, because the prevertices of such a quadrilateral will be crowded on the unit circle. (A long and narrow quadrilateral is acceptable provided that the polygon is "fat" around it, that is, if the quadrilateral can be enclosed in a disk that is mostly interior to the polygon.)

The splitting procedure has two phases. First, for every vertex v with an interior angle of $\pi/4$ or less, we chop off the corner at v as follows. Find the largest isosceles triangle T that can be formed by v and its two adjacent edges such that T is contained in P, and introduce new vertices along the two edges that are adjacent to v at the midpoints of the two sides of T. After this split, the two edges adjacent to v are said to be *protected*; that is, we do not allow them to be split during the second phase. Let P' denote the polygon obtained after this chopping phase is complete.

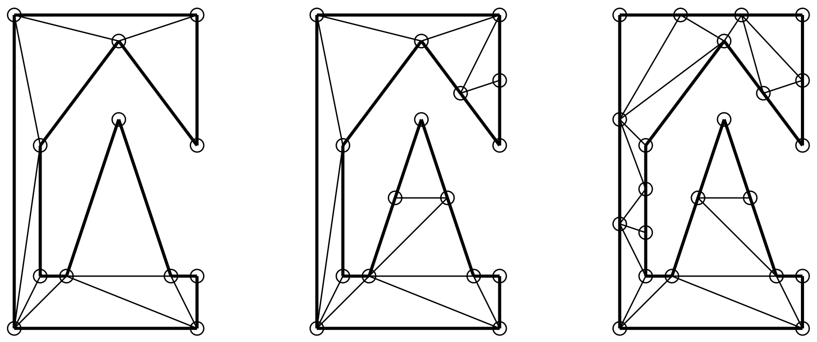

FIG. 1. *On the left is the Delaunay triangulation of a polygon. The middle shows the Delaunay triangulation after the sharp corners have been chopped in the first splitting phase. On the right is the subsequent result of the second phase, in which narrow regions are subdivided and the Delaunay triangulation is recomputed.*

The second phase of the edge-splitting procedure is iterative and generates a sequence of polygons, each of which is a subdivision of its predecessor, starting with P'. Let e be an unprotected edge of some polygon occurring in the iteration. Let $l(e)$ be its length. Let $d(e)$ be the smallest distance from e to any foreign vertex, where "foreign" means a vertex other than the endpoints of e, and distance is measured geodesically, i.e., along the shortest piecewise linear path that remains inside the polygon. (It turns out that $d(e)$ can be determined efficiently given a triangulation of the polygon.) Then we say e is *ill separated* if

$$d(e) < l(e)/(3\sqrt{2}). \tag{2}$$

At each iteration we identify all ill-separated edges and split them into three equal pieces. We repeat this until all edges are well separated. See Fig. 1 for an example of both phases of the splitting process. The splitting of edges and protecting of sharp angles is reminiscent of techniques previously introduced in the finite-element mesh generation literature; see, for example [1, 3, 4, 15]. In the mesh generation literature, the purpose of these techniques is similar to our own, namely, to prevent the occurrence of poorly shaped triangles that could arise in a triangulation of the original (unsplit) polygon. The main difference is that finite-element mesh generation subdivides the interior of the domain as well as its boundary to avoid all kinds of long, skinny quadrilaterals, whereas our splitting technique eliminates only the skinny quadrilaterals whose long edges are polygon boundaries.

We do not try to prove that the splits computed by this procedure are "effective" for our algorithm, because we do not yet have an a priori characterization of well-conditioned quadrilaterals. We do prove, however, that the second phase of the splitting procedure described in this section always terminates after a finite number of steps. Let $r(e)$ be the geodesic distance from edge e to the closest foreign edge. (A foreign edge is one that is not adjacent to e.) Let r_0 be the minimum of $r(e)$ over all unprotected edges of P'. Notice that there can be no edge shorter than r_0 in P', for if there were an edge $e_0 = (v_1, v_2)$ of length shorter than r_0, let e_1 be the other edge whose endpoint is v_1 and let e_2 be the other edge whose endpoint is v_2. Then one checks that $\mathrm{dist}(e_1, e_2) \le l(e_0) < r_0$, contradicting the choice of r_0.

We claim that the splitting procedure above never produces an edge shorter than r_0. To see this, let $e = (v_1, v_2)$ be an unprotected edge of a polygon at some intermediate stage of the above algorithm whose length is less than $3r_0$. We must argue that we could never split e. By induction, let us assume that no edges up to now are shorter than r_0. Let e_0 denote the original edge of P' that contains e. Let v be the foreign vertex closest to e, i.e., $\mathrm{dist}(v, e) = d(e)$. There are three cases: v lies on an edge that was foreign to e_0 in P'; it lies on an edge that was adjacent to e_0; or it lies on e_0 itself.

In the first case, we know that $\mathrm{dist}(e_0, v) \geq r_0$ and hence $\mathrm{dist}(e, v) \geq r_0$. But $l(e) < 3r_0$, so (2) is not satisfied and e is not split.

In the second case, let e_0' be the original edge that contains v, so that e_0 and e_0' are adjacent; say their common point is v'. Since e_0 cannot be protected by assumption, the interior angle at v' is greater than $\pi/4$. By assumption, the distance from v' to v is at least r_0. Therefore, some simple trigonometry shows that the distance from v to e_0 is greater than $r_0/\sqrt{2}$. Thus, $d(e) > r_0/\sqrt{2}$, whereas $l(e) < 3r_0$, so (2) is not satisfied.

In the third case v is collinear with e, and its distance from e again must be at least r_0, so the same reasoning shows that (2) is not satisfied.

4. Cross-ratios and embeddings.

Let a, b, c, d be four distinct points in the complex plane such that $abcd$ is a quadrilateral with counterclockwise vertex order and such that ac is an interior diagonal of the quadrilateral. We define the *cross-ratio* of these points to be

$$\rho(a, b, c, d) = \frac{(d-a)(b-c)}{(c-d)(a-b)}.$$

Note the identity $\rho(a, b, c, d) = \rho(c, d, a, b)$. Thus, the cross-ratio depends on the quadrilateral and the diagonal, but not on which endpoint of the diagonal we start at.

In general, the cross-ratio is a complex number, but there is an important special case when it is real.

LEMMA 1. *Let a, b, c, d be four distinct points on a circle in counterclockwise order. Then $\rho(a, b, c, d)$ is a negative real number.*

Proof. The angle of the quadrilateral $abcd$ at a and the angle at c are inscribed in complementary arcs of the circle, so the sum of these angles must be π. A quick diagram shows that $(d-a)/(b-a)$ has its argument equal to the angle at a, and $(b-c)/(d-c)$ has its argument equal to the angle at c. Therefore, the argument of the cross-ratio, which is the sum of these arguments, is π. □

As mentioned in the introduction, the $n-3$ primitive real variables of the nonlinear system arise from $n-3$ cross-ratios of prevertices. The preceding lemma confirms that these variables are indeed real. However, in order to implicitly impose the constraint that they be negative, our primitive variables are logarithms of the negatives of the cross-ratios (see (3) below).

We now explain *which* $n-3$ cross-ratios we use. Assume the vertices $z_1, \ldots, z_n$ of the polygon P are given in counterclockwise order. Let $d_1, \ldots, d_{n-3}$ and $Q_1 = Q(d_1), \ldots, Q_{n-3}$ be the $n-3$ diagonals and associated quadrilaterals of the Delaunay triangulation of P as defined in section 2. Let the vertices of Q_i, for $i = 1, \ldots, n-3$, be denoted by $(z_{\kappa(i,1)}, z_{\kappa(i,2)}, z_{\kappa(i,3)}, z_{\kappa(i,4)})$, so that for each i, $(\kappa(i,1), \kappa(i,2), \kappa(i,3), \kappa(i,4))$ is a four-tuple of distinct indices in $\{1, \ldots, n\}$.

Then the ith primitive variable σ_i is defined to be

$$\sigma_i = \ln(-\rho(w_{\kappa(i,1)}, w_{\kappa(i,2)}, w_{\kappa(i,3)}, w_{\kappa(i,4)})), \qquad i = 1, \dots, n-3. \tag{3}$$

It is apparent that given a list of prevertices $w_1, \dots, w_n$, the primitive variables $\sigma_1, \dots, \sigma_{n-3}$ are easily computed from (3). For evaluating the nonlinear CRDT mapping, the process must be reversed. The remainder of this section explains how to find $w_1, \dots, w_n$ on the unit circle to satisfy (3) given $\sigma_1, \dots, \sigma_{n-3}$.

Notice that there are three degrees of freedom, because (3) imposes only $n-3$ real constraints on n real variables. We will use the flexibility afforded by these degrees of freedom to our advantage; indeed, they are the reason that the CRDT algorithm avoids problems with crowding, as discussed in section 6. For now, let us fix these degrees of freedom by assuming that the three prevertices corresponding to a Delaunay triangle T_0 are arbitrarily placed on the unit circle in a manner preserving their ordering. (Later, we will show that the choice of T_0 and the three prevertex positions will not affect the S–C image. See Theorem 2 at the end of this section.)

We now show how to embed the remaining $n-3$ vertices using the cross-ratio information, starting with a lemma that tells us how to place a single prevertex.

LEMMA 2. *Given distinct points a, b, c on the unit circle in counterclockwise order, and given a negative real number ρ_0, there exists a unique point d on the unit circle such that $\rho(a, b, c, d) = \rho_0$. Furthermore, this point is counterclockwise from c and clockwise from a.*

Proof. If we write out the formula and substitute, we get an explicit closed formula for d,

$$d = \frac{hc + a}{h + 1}, \tag{4}$$

where

$$h = \frac{\rho_0(b - a)}{(c - b)}.$$

We must first show that $h \neq -1$ so that the denominator in the formula for d is nonzero. But this is obvious, because $(b - a)/(c - b)$ must have a nonzero imaginary part (since a, b, c cannot be collinear), so h also has a nonzero imaginary part. Thus d is uniquely determined.

Next, we must show that d lies on the unit circle between c and a. Consider sliding a test point along the unit circle starting from very near a clockwise to c. It is easy to check that the cross-ratio, which is a negative real number by the earlier lemma, varies continuously from 0 to $-\infty$. Therefore, its value must be ρ_0 at some intermediate point. But the last paragraph shows that this intermediate point is unique. $\square$

Remark. If the distances between a, b, and c become very small, we may not be able to compute d with high precision relative to these small distances. However, we can find d accurately relative to well-separated points on the circle, which is all CRDT requires.

Now the first main theorem of this section tells us that, assuming the first three prevertices are determined, we can place the remaining $n-3$ prevertices uniquely.

THEOREM 1. *Let T_0 be any triangle in the Delaunay triangulation of P, and let its vertices in counterclockwise order be indexed as z_ϕ, z_ψ, z_χ. Suppose the prevertices w_ϕ, w_ψ, w_χ are specified as distinct points on the unit circle in counterclockwise order.*

Then, given any real-number values for the primitive variables $\sigma_1, \ldots, \sigma_{n-3}$, there exists a unique solution to (3)*, that is, a unique way to define the remaining $n-3$ w_i's on the unit circle satisfying* (3)*. The algorithm to find the w_i's satisfying* (3) *is linear time. Furthermore, for this solution to* (3)*, $w_1, \ldots, w_n$ will be in the correct counterclockwise order. We call such a placement of the prevertices an* embedding*.*

Proof. This proof relies heavily on the fact that the dual of the Delaunay triangulation is a tree. Let T_1 be a triangle sharing a diagonal with T_0. The two vertices of T_1 shared with T_0 are already embedded, and we can embed T_1's third vertex using (4) and the given value σ_i, where i is the index of the quadrilateral $T_0 \cup T_1$. Using this rule to position the third vertex of T_1 ensures that (3) holds for this particular i. Now that the embedding of T_1 is known, we can embed T_1's neighbors and so on. The general rule is as follows: regard the dual tree as being rooted at T_0, inducing a parent-child relation on the Delaunay triangles. We can embed each triangle after its parent is embedded, starting with the known embedding of T_0. The edges of the dual tree are in one-to-one correspondence with quadrilaterals in the Delaunay triangulation, so we use each given σ_i exactly once in this process. This shows that (3) will be satisfied for $i = 1, \ldots, n-3$. Furthermore, an induction argument shows that this embedding is uniquely determined because every step of this construction is forced.

Finally, we must show that the w_k's end up in the correct counterclockwise order. Consider embedding T_j with prevertices w_a, w_b, w_c, and suppose that w_a, w_b are already in place. Lemma 2 ensures that w_c will be in the arc between w_a, w_b complementary to the arc that contains the parent triangle's other prevertex. Therefore, w_c is placed correctly with respect to w_a and w_b. But notice that w_c must be the first prevertex placed between w_a and w_b because any other prevertices on this arc arise from children of T_j. Thus, the placement of w_c is correct with respect to all vertices placed before it. ▯

This theorem shows that given values for the primitive variables and the positions of the three prevertices corresponding to a Delaunay triangle, we can compute the positions of all of the prevertices. How should we choose the initial triangle and embed its prevertices? It turns out that *any* initialization is acceptable; all embeddings give the same image polygon, up to similarity transform.

To show this, we begin by recalling some standard facts from complex analysis [16]. First, any conformal map from the unit disk to itself is a fractional linear transformation of the form

$$g(z) = e^{i\theta} \frac{z - r}{1 - \bar{r}z}, \tag{5}$$

where r is a complex number such that $|r| < 1$ and $\theta \in [0, 2\pi)$. As a consequence, there is a unique such mapping that takes any three distinct points on the unit circle to any other three points on the unit circle, preserving ordering [16]. We formalize another consequence in a lemma.

LEMMA 3. *Let a, b, c, d be four points on the unit circle in counterclockwise order. Let g be a conformal mapping of the unit disk to itself. Then*

$$\rho(a, b, c, d) = \rho(g(a), g(b), g(c), g(d)).$$

Proof. Simple algebra verifies that cross-ratios are invariant under fractional linear transformations. See [16] for details. ▯

We now show that no matter which initial triangle T_0 we select in Theorem 1, and no matter how we choose the positions of its three prevertices, we end up with the same image polygon, up to similarity transform.

THEOREM 2. *Let $\sigma_1, \ldots, \sigma_{n-3}$ be given. Let T_0, T_0' be two triangles in the Delaunay triangulation whose vertices are (z_ϕ, z_ψ, z_χ) and $(z_{\phi'}, z_{\psi'}, z_{\chi'})$, respectively. Let (w_ϕ, w_ψ, w_χ) and $(w'_{\phi'}, w'_{\psi'}, w'_{\chi'})$ be order-preserving embeddings of their prevertices in the unit disk as in Theorem* 1. *Use the construction of Theorem* 1 *to produce the two embeddings $(w_1, \ldots, w_n)$ and $(w'_1, \ldots, w'_n)$, respectively. Let $\tilde{P}$, $\tilde{P}'$ be the images of the unit disk under* (1) *using these two prevertex sets. Then $\tilde{P}$ and $\tilde{P}'$ are* similar; *i.e., they agree up to a translation, rotation, and scaling.*

Remark. We have not explained how to choose the affine constants A and B in (1). The theorem holds for any choice of these constants. Alternatively, the theorem asserts that, given the constants for one embedding, these constants can be chosen for the other embedding in such a way that the image polygons coincide.

Proof. Let $(w'_\phi, w'_\psi, w'_\chi)$ be the positions of the prevertices of T_0 in the second embedding. Let g be the conformal mapping of the unit disk to itself carrying the points (w_ϕ, w_ψ, w_χ) to $(w'_\phi, w'_\psi, w'_\chi)$. That is, g maps the prevertices of T_0 in one embedding to the prevertices of T_0 in the other.

We claim that $g(w_i) = w'_i$ for *all* i, not just $\{\phi, \psi, \chi\}$. Because g preserves cross-ratios, the embedding $(g(w_1), \ldots, g(w_n))$ has the same $n-3$ cross-ratios as $(w_1, \ldots, w_n)$, which by assumption has the same $n-3$ cross-ratios as $(w'_1, \ldots, w'_n)$. But since three entries in $(g(w_1), \ldots, g(w_n))$ are equal to the three corresponding entries in $(w'_1, \ldots, w'_n)$, the uniqueness part of Theorem 1 guarantees that $g(w_i) = w'_i$ for all i.

Now consider composing (1) with the conformal mapping g^{-1} of the disk to itself. The composition is a conformal mapping from the unit disk to $\tilde{P}$, so the S–C formula (1) must also hold when the prevertices are given by $(w'_1, \ldots, w'_n)$ for a suitable choice of the affine constants. So $\tilde{P}$ and $\tilde{P}'$ are similar. □

5. The CRDT algorithm.

We are now prepared to specify the CRDT algorithm in more detail.

Step 1. **Split the edges of the polygon as described in section 3.** We again use P to denote the polygon obtained after splitting. Let n be the number of its vertices.

Step 2. **Compute the Delaunay triangulation of P.** Now that the Delaunay triangulation is computed, we can fix a particular numbering of the diagonals and quadrilaterals in the triangulation. Recall the notation $\kappa(i,j)$ used in (3) to number the vertices of the quadrilaterals.

Let us define

$$c_i = \ln(|\rho(z_{\kappa(i,1)}, z_{\kappa(i,2)}, z_{\kappa(i,3)}, z_{\kappa(i,4)})|) \tag{6}$$

for $i = 1, \ldots, n-3$. Note that the cross-ratio in this formula in general will be a complex number, so the absolute value symbols denote magnitude.

Step 3. **Solve the nonlinear system $\mathbf{F}(\boldsymbol{\sigma}) = \mathbf{0}$.** The map $\mathbf{F} : \mathbf{R}^{n-3} \to \mathbf{R}^{n-3}$ is defined as follows. The input variables are the primitive variables $\sigma_1, \ldots, \sigma_{n-3}$ defined by (3). It was shown in section 4 that there is a unique (up to similarity) S–C mapping that can be computed from these primitive variables. Let $\zeta_1, \ldots, \zeta_n$ be the vertices of the image of (1). For $i = 1, \ldots, n-3$, let

$$F_i(\sigma_1, \ldots, \sigma_{n-3}) = \ln(|\rho(\zeta_{\kappa(i,1)}, \zeta_{\kappa(i,2)}, \zeta_{\kappa(i,3)}, \zeta_{\kappa(i,4)})|) - c_i.$$

Observe that although the ζ's themselves are determined only up to similarity transform, the cross-ratio of four of them is invariant under similarity transform, so this definition makes sense.

For each i we need to find the four image vertices $\zeta_{\kappa(i,1)}, \ldots, \zeta_{\kappa(i,4)}$. To do so we construct a certain embedding E_i of the prevertices. Recall from section 4 that given the vector $\boldsymbol{\sigma}$, we can arbitrarily embed three of the prevertices, such as $w_{\kappa(i,1)}, w_{\kappa(i,2)}, w_{\kappa(i,3)}$. We embed these three in such a way that when $w_{\kappa(i,4)}$ gets placed, these four prevertices will be arranged in a rectangle centered at the origin with the correct cross-ratio. Then, as in Theorem 1, we position the rest of the prevertices to complete E_i. This embedding defines an S–C map f_i, given by (1) with $A = 0$ and $B = 1$, which we use to compute the relative positions of the four image vertices $\zeta_{\kappa(i,1)}, \ldots, \zeta_{\kappa(i,4)}$. The path of integration is a straight-line segment from the origin, and we use the compound Gauss–Jacobi quadrature rule described by Trefethen [18].

We discuss nonlinear solvers in more detail in section 7.

Note that at a solution to $\mathbf{F}(\boldsymbol{\sigma}) = \mathbf{0}$, we have for $i = 1, \ldots, n-3$ that

$$|\rho(\zeta_{\kappa(i,1)}, \zeta_{\kappa(i,2)}, \zeta_{\kappa(i,3)}, \zeta_{\kappa(i,4)})| = |\rho(z_{\kappa(i,1)}, z_{\kappa(i,2)}, z_{\kappa(i,3)}, z_{\kappa(i,4)})|.$$

Furthermore, we know that all the interior angles of the polygon determined by $\zeta_1, \ldots, \zeta_n$ are correct because the angles are inherent in the S–C mapping. Is this polygon the correct one? We state the result as a proposition, whose proof has been communicated to us by Snoeyink [17].

PROPOSITION 1. *Let P be an n-vertex triangulated simple polygon. Then P is uniquely determined up to similarity transform by the following data:*

1. *the sequence of all interior angles of P at its vertices, and*
2. *the list of $n-3$ absolute values of cross-ratios of the quadrilaterals determined by the triangulation of P.*

We could have chosen the $n-3$ equations as the conditions on side lengths used by SCPACK and the SC Toolbox. The advantage of equations that enforce cross-ratio conditions is that the system of nonlinear equations apparently has a desirable monotonicity property, described in section 7.

Step 4. **Compute the constants in the affine transformations.** When $\mathbf{F}(\boldsymbol{\sigma}) = \mathbf{0}$, we know that the map f_i applied to the prevertices of quadrilateral Q_i in embedding E_i produces a quadrilateral Q_i' that is similar to Q_i. Two points of each quadrilateral define the transformation between them, so we can solve either a linear system or a least-squares problem to find the constants A_i and B_i that appear in (1). This completes the map from E_i to P.

In section 9 we describe an alternative method, needed in certain situations, for computing the affine constants. $\square$

Observe that CRDT essentially computes $n-3$ simultaneous S–C maps from the disk to P. In the following sections we will see how each map is locally accurate in a region of P that overlaps with other such local regions.

6. On the circumvention of crowding.

In this section we explain the crux of our claim that CRDT is unaffected by crowding. If the domain P contains any long, narrow channel, then for any possible correct embedding of the prevertices, some of the prevertices will be extremely crowded. But the embedding E_i (described at the end of the last section) for computing the ith component of $\mathbf{F}$ guarantees that $w_{\kappa(i,1)}, \ldots, w_{\kappa(i,4)}$ will *not* be crowded, either against each other or against any other prevertex. Therefore, the crowding has no impact on the accuracy of the quadrature rule applied to these four prevertices, because the path of integration never passes close to crowded prevertices. Thus, it is the flexibility to re-embed the prevertices for each coordinate entry of $\mathbf{F}$, along with the splitting of narrow channels, that allows us to circumvent crowding. See Fig. 2.

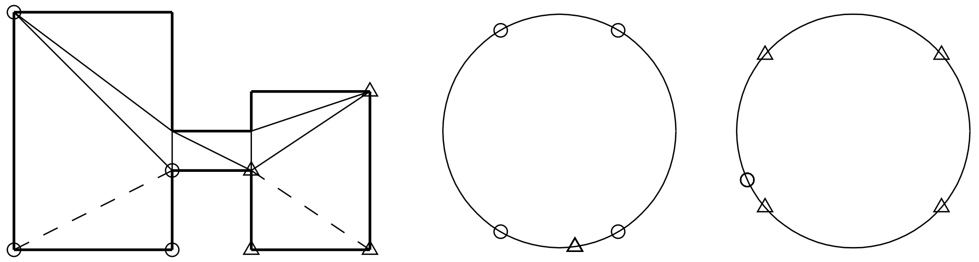

FIG. 2. *On the left is a triangulated polygon with two distinguished quadrilaterals whose diagonals are given as dashed lines. No embedding of the S–C prevertices will be globally uncrowded. In the middle picture we see an embedding that keeps the prevertices of the first quadrilateral (marked with circles) uncrowded, while the prevertices of the other (marked with triangles) are crowded—too closely to be distinguished. However, in another embedding (far right), the crowding situation is reversed.*

In order to confirm that none of the four prevertices are crowded against each other or their neighbors, we need a result stating that none of the cross-ratios $\rho_1, \dots, \rho_{n-3}$ of the prevertices is very large (close to $-\infty$) or very small (close to 0). Unfortunately, there cannot exist fixed (constant) upper or lower bounds on these cross-ratios that apply to all polygons, as a regular n-gon shows. Any triangulation of the regular n-gon is a Delaunay triangulation, so CRDT might compute a triangulation that has a quadrilateral whose aspect ratio is $O(n)$. Since the S–C mapping of an n-gon is close to the identity mapping, there will also be four prevertices whose cross-ratio is $O(n^2)$, or $O(n^{-2})$. Thus, there is no a priori upper or lower bound possible on the ρ_i's, and therefore none on the σ_i's either. This growth of the σ_i's is very slow (logarithmic in n) and is thus not expected to have a significant impact on the accuracy of CRDT. But the absence of a constant upper bound or lower bound means, for example, that there is no a priori upper bound on how much adaptation is necessary in the compound Gauss–Jacobi integration used to evaluate (1).

A more plausible conjecture might be that the difference $\sigma_i - c_i$ (where c_i was defined by (6)) has constant upper and lower bounds. This result would show that CRDT is not affected by crowding in the sense that it never works with distances that are substantially shorter than side lengths in the original polygon. Note that both quantities σ_i and c_i are logarithms, so subtracting them is the right way to check how they differ. We do not know whether this conjecture is true, but we have not seen substantial divergence between σ_i and c_i in our computational experiments. In Fig. 3 we present a histogram of $\sigma_i - c_i$ for all polygons in our experiments, for all indices i, at the final solutions obtained by the CRDT algorithm. Notice that points in this histogram lie in a fairly narrow range. (The cross-ratios stay similarly bounded in intermediate steps of the iterations because of the monotonicity observed in our nonlinear equations, described in the next section.)

The representation of prevertices used by the SC Toolbox and SCPACK is also well conditioned in the presence of crowding, because logarithms of distances between prevertices are used as primitive variables rather than the prevertex positions themselves. But the logarithmic distances cannot be used directly to compute S–C maps, so both packages pass (in an intermediate step) from these primitive variables to a particular prevertex embedding used to evaluate all the components of $\mathbf{F}$, causing some of them to be affected by cancellation due to crowding. By contrast, the well-

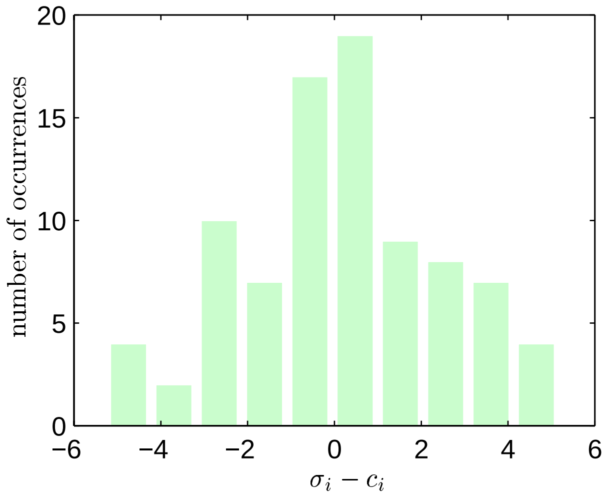

FIG. 3. *Evidence of controlled cross-ratio sizes. The histogram depicts the discrepancies $\sigma_i - c_i$ between the primitive cross-ratio variables and polygon cross-ratios for the experiments in section 7. The variables are logarithmic. The clustering near zero and absence of large outliers supports the claim that CRDT circumvents crowding.*

conditioned cross-ratio representation used in CRDT is transformed into separate embeddings for each component of $\mathbf{F}$, so that the prevertices of interest are never crowded and cancellation is avoided.

Does this mean that CRDT "solves" all problems with crowding? If one takes the results of CRDT and computes a single, fixed embedding of the prevertices, then there is no improvement compared to the Toolbox with regard to crowding. Therefore, if one wants to use CRDT in an application, then one must devise an algorithm in which the cross-ratios produced by CRDT are used to shuffle between different embeddings of the prevertices. We give some examples of problems from potential theory that can be solved in this manner by CRDT in section 9. In fact, for every reasonable problem in potential theory that we have considered, we have always been able to come up with a way to use the cross-ratios directly and avoid crowding. But in each case the technique we have devised is slightly different, so we are not able to state with certainty that CRDT can solve all problems in potential theory posed on elongated polygons.

7. Computational experiments with CRDT. In this section we experiment with an implementation of CRDT in MATLAB. We present some evidence of the monotonicity of the nonlinear system and consider the matter of solving it numerically. We also compare the performance of the CRDT algorithm to the SC Toolbox for MATLAB [7] and find that CRDT is competitive for most regions. The principal exceptions are regions which cause the edge-splitting algorithm to add a great many extra vertices of angle π. While such vertices do not affect the amount of work in computing the S–C integral (1), they do affect the size of the nonlinear system to be solved.

The computational core of CRDT is solving the nonlinear system $\mathbf{F}(\boldsymbol{\sigma}) = \mathbf{0}$. We consider nonlinear solvers that require only function evaluations. According to our experiments, a particular very simple linear iteration always converges at a linear

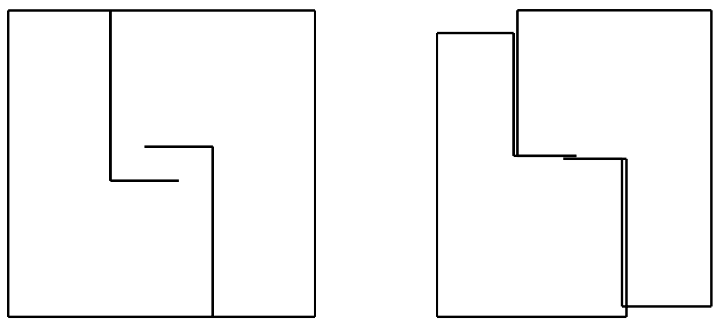

FIG. 4. *Lack of global convergence in the SC Toolbox. In solving for the polygon on the left,* `dparam` *arrives at the polygon on the right and terminates. The horizontal slits cannot move past each other without temporarily increasing the nonlinear system residual.*

rate. This iteration starts with $\boldsymbol{\sigma}^{(0)} = \mathbf{c}$, where c_i was defined above by (6). Then we iterate:

$$\boldsymbol{\sigma}^{(k+1)} = \boldsymbol{\sigma}^{(k)} - \mathbf{F}(\boldsymbol{\sigma}^{(k)}). \tag{7}$$

Our conclusion from experiments is that this iteration always satisfies $||\mathbf{F}(\boldsymbol{\sigma}^{(k+1)})||_2 \leq \alpha||\mathbf{F}(\boldsymbol{\sigma}^{(k)})||_2$, for an α that is problem-dependent but always satisfies $\alpha < 1$. We have not been able to come up with a convincing explanation for this behavior. The essential reason is apparently that the Jacobian $\mathbf{F}'$ approximates the identity, but none of the likely conditions on $\mathbf{F}'$ that would support this claim have been found to hold in experiments.

The smooth convergence of (7) makes us suspect that $\mathbf{F}$ has some strong monotonicity property. Expected consequences of this monotonicity are that $\|\mathbf{F}\|$ has no local minima and that $\mathbf{F}$ is injective. In contrast, the nonlinear system used by SCPACK and the SC Toolbox does not have strong monotonicity properties and in fact is prone to local minima [9]. This is demonstrated by Fig. 4. If the SC Toolbox is used to solve for the polygon on the left in the figure, it arrives at the polygon on the right and terminates due to an apparent local minimum in the system. The minimum results from the fact that the two horizontal slits cannot be moved past each other without temporarily increasing the solution residual. This phenomenon was first described by Howell [9], who also points out that crowding often masks the effect. However, in this case the correct prevertices are not fatally crowded in double precision, and the Toolbox will find the solution if given a good enough starting guess. We cannot be certain that there do not exist polygons for which CRDT exhibits similar global convergence difficulties.

In practice, the convergence of the simple iteration (7) is too slow. Instead we use the nonlinear equation solver NESOLVE due to Behrens, which is based on a Gauss–Newton method with a Broyden update of $\mathbf{F}'$, as described in [6]. This is the same nonlinear system package used by the SC Toolbox. In one variant, Full CRDT, we use the standard finite-difference Jacobian to seed the Broyden update. In the other, Shortcut CRDT, we attempt to exploit the monotonicity by setting the initial $F' = I$. In Fig. 5 we show convergence curves for a typical case, the goblet region in Fig. 6. The linear convergence of (7) is strikingly smooth. The convergence of the NESOLVE variations is more complex, but overall is approximately linear at much

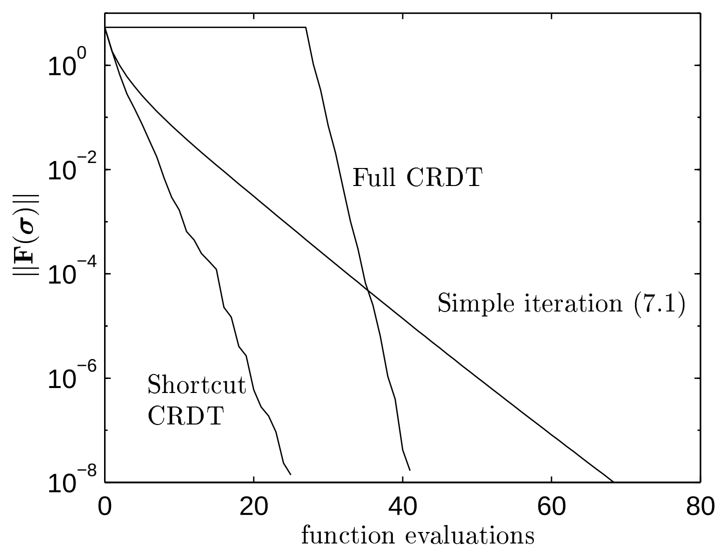

FIG. 5. *Convergence curves for numerical solutions of* $\mathbf{F}(\boldsymbol{\sigma}) = \mathbf{0}$. *The iteration* (7) *converges linearly. The NESOLVE variations exhibit approximately linear convergence as well, but at much better rates.*

better rates than the simple iteration. The slower rate for Shortcut CRDT is more than offset by the savings gained by not initializing $\mathbf{F}'$.

We compare the performance of CRDT to the SC Toolbox functions `dparam` or `rparam` for maps from the disk or rectangle, respectively, depending on whether the target region is elongated. From each method we demand a nonlinear residual with maximum norm no larger than 10^{-8}. All the experiments reported here were performed on a SPARCstation-10.

Figure 6 shows four experimental polygons with their Delaunay triangulations, after the edge splitting has been done. In Table 1 we present the number of function evaluations and total CPU time required by the nonlinear system solver for these regions.

TABLE 1
Performance of CRDT variants versus the SC Toolbox for the solution of the parameter problem for the regions in Fig. 6. *The first number in each entry is the number of nonlinear function evaluations made by the nonlinear solver; the other number is the CPU time in seconds.*

	SC Toolbox	Full CRDT	Shortcut CRDT
Cross	31/10.03	18/14.52	12/11.78
Goblet	78/14.35	42/142.3	26/112.3
Spiral	186/275.4	51/237.1	34/197.0
Y	−/−	35/76.37	25/70.38

Note that Shortcut CRDT is indeed always faster than Full CRDT. Also observe that the individual SC Toolbox iterations are much faster than those for CRDT. This is because of the additional unknowns introduced by splitting and the multiple embeddings used by CRDT to thwart crowding. However, the nonlinear systems posed by the SC Toolbox for these polygons are not as easily solved, and thus many more iterations may be required even though the systems are smaller.

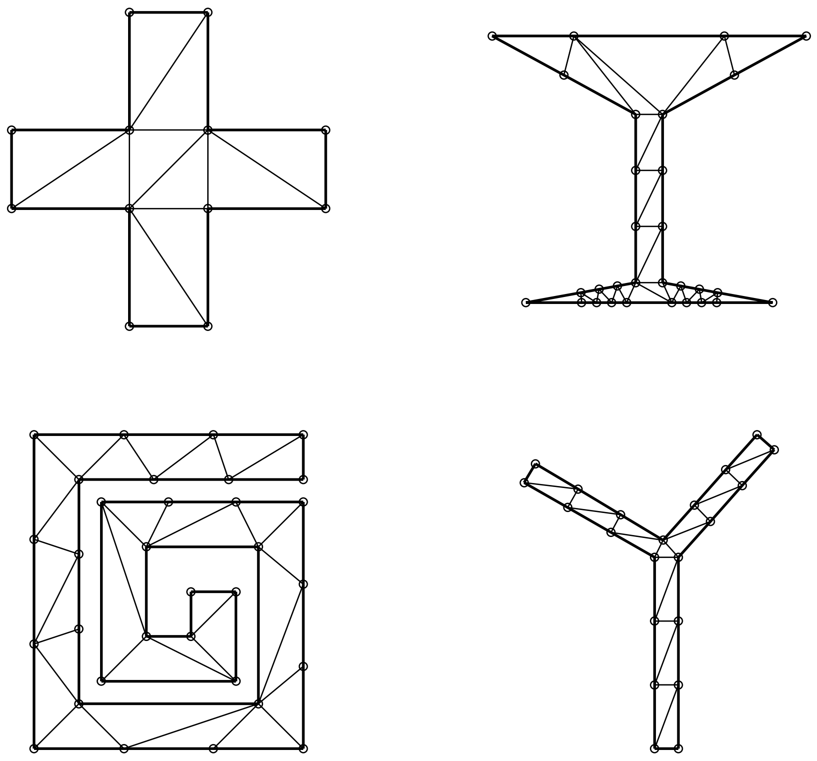

FIG. 6. *Four regions on which CRDT experiments were performed, after splitting and Delaunay triangulation. See text.*

The cross-shaped region (top left) is not elongated and both `dparam` and the CRDT variants converge rapidly. The SC Toolbox has less overhead and `dparam` is slightly faster. The goblet region (top right) has 22 vertices added by the splitting algorithm to its original eight. These extra vertices greatly slow down the CRDT solvers, making `dparam` much faster even though it has some difficulty finding the solution. For the spiral (bottom left), the solution is sufficiently difficult for `rparam` that CRDT is a little faster. Finally, for the Y-shaped region (bottom right), the SC Toolbox is unable to find a solution because of the doubly elongated nature of the region. CRDT, however, finds the solution easily.

These examples demonstrate what we observe about CRDT in general. CRDT is least efficient when many extra vertices are added during splitting. This most often occurs near sharp corners and in narrow channels of the region, both of which are prominent in the goblet region. While it seems that there is no way to circumvent subdividing a channel because of crowding effects, we do not know if there is a more efficient way to produce well-conditioned quadrilaterals near sharp corners. On the other hand, CRDT handles multiply elongated regions with no difficulty, something which previously no general-purpose method for S–C mapping has been able to do.

8. Computing maps. In section 6 we remarked that in order to use the information obtained by CRDT in the parameter problem solution, we would have to continue to work with multiple simultaneous embeddings of the prevertices. In this section we illustrate this idea by describing how to map points from the disk.

We set the stage by describing another sort of dual to the Delaunay triangulation of polygon P. We define the *quadrilateral graph* to have a node for each of the $n-3$ quadrilaterals in P, with nodes i and j being adjacent when Q_i and Q_j share three vertices. It is easy to see that the quadrilateral graph is connected.

The significance of the quadrilateral graph is in the overlapping of adjacent quadrilaterals. Embedding E_i is assumed to give an S–C map that is locally accurate in Q_i. If Q_i is adjacent to Q_j, they share three vertices and hence three prevertices. Thus we can find the fractional linear transformation between E_i and E_j. Moreover, since a quadrilateral's prevertices are well separated in its own embedding, this transformation can be computed very accurately. This is not necessarily the case for embeddings that are not adjacent, as crowding could introduce fatal ill conditioning.

The first step in mapping from the disk is to choose a particular reference embedding of the disk in which to specify points. For this purpose we will assume that $w_n = 1$ and that the conformal center $z_0 = f(0)$ is given; this fixes the three degrees of freedom and implicitly selects an embedding of the prevertices. Let Q_k be a quadrilateral of P that contains z_0. We can invert the affine transformation defined by A_k and B_k and invert the S–C integral f_k numerically to find the inverse image w_0 of z_0 in embedding E_k.

Because E_k yields a locally accurate map for Q_k, we expect that w_0 will not be too close to the boundary of the disk. Thus, given a point w in the reference embedding, we can find its image under transformation to embedding E_k. Now we can accurately map w to all the embeddings adjacent to E_k in the quadrilateral graph. We can continue this process iteratively until the equivalents of w are known in all embeddings. Any embedding can in principle be used to find the image of w. However, because we expect each local map to be most accurate near the origin, we select the embedding in which the equivalent of w is smallest in magnitude.

Inverse mapping is similar. Given a point z in P, we invert z numerically in an embedding corresponding to a quadrilateral containing z. We then transform this inverse image through the quadrilateral graph to embedding E_k and thence to the reference embedding.

9. Applications.

In this section we describe two applications of CRDT that demonstrate its ability to handle crowding in practice. While we cannot specify a universal recipe, we believe that the techniques of this section can be adapted to suit a variety of situations. In both techniques we rely heavily on the quadrilateral graph described in section 8.

Our first example is to compute the map from a rectangle to a *generalized quadrilateral*. For our purposes, a generalized quadrilateral is a polygon with four distinguished vertices, which map to the four corners of a rectangle. The fact that four, rather than three, of the vertices are constrained is compensated by the fact that the aspect ratio of the rectangle must be a certain unknown value, known as the *conformal modulus*. Computation of this mapping is equivalent to solving Laplace's equation on the polygon with the Dirichlet values of 0 and 1 on two generalized sides separated by two generalized sides with homogeneous Neumann conditions.

For a multiply elongated region, branches other than the main channel will collapse into crowded clusters on the sides of the rectangle. Computing the map accurately in the vicinity of these clusters (into the collapsed branches) is therefore challenging.

In Fig. 7 we plot the images of straight lines in a rectangle to a certain quadrilateral. The rectangle has a width of 1 and a height of about 18.2, which is the conformal

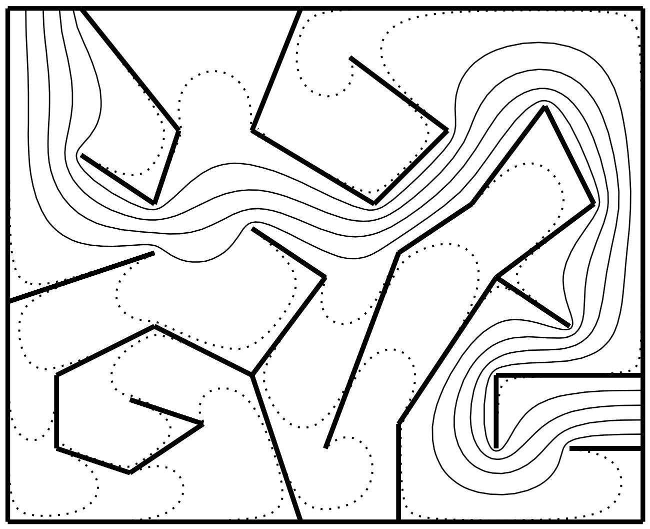

FIG. 7. *Rectangle map to a "maze." The exits of the maze map to the short rectangle sides, which are normalized to unit length. The maze has several branches that are very crowded on the long rectangle sides. The solid curves in the maze are images of lines with abscissae* 0.2*,* 0.4*,* 0.6*, and* 0.8*. The preimages of the dotted curves are separated from the long rectangle edges by* 10^{-2}*,* 10^{-4}*, ...,* 10^{-16}*. All computations were performed in double precision.*

modulus. The solid curves are images of lines which are well separated from the long edges of the rectangle, so they stay out of the side turns, or "deadwaters." The dotted curves have preimages that are exponentially close to the long rectangle edges, and they closely follow the maze boundary into the deadwaters. The crowding of the spiral branch is comparable to machine precision, and it would pose no difficulty if it were much more crowded. As far as we know, no other conformal mapping algorithm can accurately compute these curves.

We proceed to describe how we produced Fig. 7. Suppose a valid embedding of prevertices for the S–C map to polygon P is known. Define a new vector of turning angles $\tilde{\beta}$, where $\tilde{\beta}_j = -1/2$ if vertex j is distinguished and $\tilde{\beta}_j = 0$ otherwise. We define a new S–C map by using $\tilde{\beta}$ in place of β in (1) and the same prevertices. We call this new map the *rectified map* for the polygon, because the image of the disk under this map is clearly a rectangle. Thus the composition of the inverse of the rectified map with the original map is the desired rectangle map.

In order to compute the map between the rectangle and embedding E_i, we must find the appropriate affine constants $\tilde{A}_i$ and $\tilde{B}_i$ that follow the S–C integrals $\tilde{f}_i$. This process is complicated by the fact that the vertices of the rectangle are unknown; we do not even know the conformal modulus. Notice that (allowing degenerate triangles) we can define the same triangulation and quadrilateral structure on the rectangle as on P.

The rectangle is determined only up to a similarity, so let a reference side e be given. Choose a quadrilateral $\tilde{Q}_k$ that has e as an edge. As before, we compute $\tilde{f}_k$ on the prevertices of $\tilde{Q}_k$ to find $\tilde{Q}'_k$. Find $\tilde{A}_k$ and $\tilde{B}_k$ directly, by comparing $\tilde{Q}'_k$ to e. Now visit a neighboring quadrilateral $\tilde{Q}_j$. Since $\tilde{Q}_j$ shares three vertices with $\tilde{Q}_k$, we can find the affine transformation that relates $\tilde{Q}_j$ to $\tilde{Q}_k$. Furthermore, this is a well-conditioned step because of the local accuracies of the embeddings. But then by composition with the known affine constants $\tilde{A}_k$ and $\tilde{B}_k$, we can find $\tilde{A}_j$ and $\tilde{B}_j$. The moduli of all the $\tilde{B}_i$ may range over many orders of magnitude, but the above process leads to each by multiplication, rather than subtraction, so high relative accuracy is maintained.

As the affine transformations to the rectangle are computed, they can be used to compute the images of the vertices of P on the rectangle. The rectangle prevertices will all be found with high accuracy relative to the overall size of the rectangle, and the conformal modulus can be found accurately as well.

To produce Fig. 7, we ran Shortcut CRDT on the polygon, which after splitting had 87 vertices. This found the solution vector $\boldsymbol{\sigma}$ to a residual tolerance of 10^{-8} after just 27 function evaluations. Then we computed the affine constants for the standard and rectified maps, and the rectangle prevertices. Now suppose the rectangle line we wish to map is separated from the rectangle boundary by a distance h. In the image plane of $\tilde{f}_i$, that separation scales to $h/|\tilde{B}_i|$. If $h/|\tilde{B}_i| \ll 1$, the image of the line will be very close to the boundary of P in the vicinity of Q_i. If $h/|B_i| \gg 1$, then $\tilde{Q}_i$ is degenerate, Q_i is in a deadwater region, and the image of the line will be far from Q_i. In short, only for those embeddings in which h is comparable to $|\tilde{B}_i|$ do we need to track the image of the line. In these cases we numerically invert $\tilde{f}_i$ and apply f_i and then (A_i,B_i).

We believe that this technique can be generalized to perform grid generation for any polygon. Instead of mapping from a rectangle, one would map from a more general region whose angles are integer multiples of $\frac{\pi}{2}$. It is trivial to find an orthogonal grid in such a region, because all edges could be aligned with coordinate axes. The main obstacle is in automatically assigning the most appropriate angle at each prevertex.

Our second example of applying CRDT involves the solution to a certain boundary value problem related to harmonic measure. Our BVP is Laplace's equation $\triangle u = 0$ with Dirichlet boundary conditions on a polygonal domain P. There is one edge s of the polygon, which we call the *forced edge*, with boundary condition 1, and the solution on the rest of the boundary is zero. We want to find u only at a particular point x in the interior of P. Let the endpoints of s be denoted z_p and z_q, in counterclockwise order.

Mathematically the problem is easily solved. Let $f_0 : D \to P$ be an S–C map that has $f_0(0) = x$. The solution u of the original problem maps to a Laplace solution $\hat{u}$ on the unit disk via the formula $\hat{u}(w) = u(f_0(z))$. Because the solution to Laplace's equation at the center of a disk is identically the average of the boundary Dirichlet values, we have that $u(x)$ is precisely the arc-length distance θ measured in radians between $f_0^{-1}(z_p)$ and $f_0^{-1}(z_q)$, scaled by $1/(2\pi)$.

An obvious algorithm is to compute the prevertices of the disk map and read off the angular distance between prevertices p and q. In fact, it suffices to compute $h = |f_0^{-1}(z_p) - f_0^{-1}(z_q)|$. For, given the Euclidean distance h between two points on the unit circle, we can get the angular distance via

$$\theta = 2\sin^{-1}(h/2). \tag{8}$$

However, this direct method will fail when the forced edge s is separated from x by a long, thin region, as in Fig. 8. In this case, the value $u(x)$ will be extremely close to zero, and cancellation error will dominate h. Yet we insist on computing $u(x)$ accurately in a *relative* sense. It suffices to compute h with high relative accuracy, since (8) can be evaluated accurately when h is very small (indeed, $\theta \approx h$ for small h).

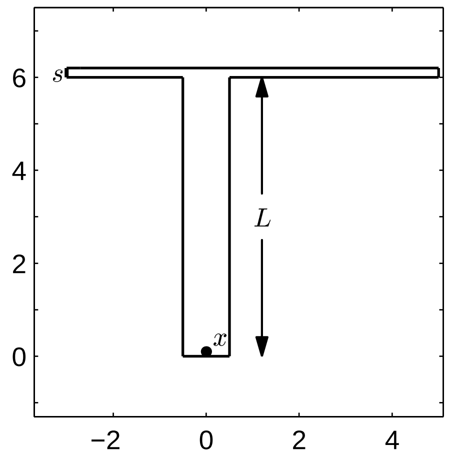

FIG. 8. *The BVP of section* 9 *is solved on this T-shaped region. The length L is allowed to vary, and the forced edge is denoted s. The point x at which the solution is sought is at* $0.1i$. *The dimensions of the horizontal bar are* 8×0.2.

If the domain P is singly elongated, a solution to the BVP is possible using an SC Toolbox map from a rectangle. A rectangle is transformed to the upper half-plane by the elliptic function $\operatorname{sn}(z|m)$, where m is obtained in the S–C solution. We can further construct a fractional linear transformation that maps the half-plane image of x to the center of the unit disk, and then accurately measure h. Of course, such a solution is not available when P is multiply elongated and the Toolbox rectangle map fails.

We assume throughout that x is not too close to any edge of the polygon. If x is close to an edge, this creates a different problem with relative accuracy that is not addressed here, though it could be addressed by other means.

We start by running CRDT on P and computing the affine transformation constants. Let Q_1 be a quadrilateral having s as an edge. (We assume for now that s is an unsplit edge of the original polygon and deal with the case of split s below.) Let Q_m be a quadrilateral in P that contains the point x. We can accurately invert the S–C map using embedding E_m to find ξ. Furthermore, ξ will not be close to the boundary of the disk.

Let $Q_1, Q_2, \ldots, Q_m$ be a path in the quadrilateral graph. In the embedding E_2, we can compute the distance $|w_p - w_q|$ accurately, because either w_p or w_q is a vertex of Q_2. Thus we can compute the cross-ratio ρ_2 of w_p, w_q, and two more prevertices of Q_2, including the prevertex that is not shared with Q_1. Now consider the embedding E_3. In this embedding, w_p and w_q may be crowded. However, we still know the cross-ratio ρ_2 of w_p, w_q, and two prevertices of Q_3. Hence, of the four lengths that are factors in the formula for $|\rho_2|$, only one may be small, so we can compute it accurately using ρ_2 and the other lengths. By the same token, we can accurately compute ρ_3, the cross-ratio of w_p, w_q, and two other prevertices of Q_3, including the one not shared by Q_2. As in the previous example, we are computing a very small quantity (given by (8)) by repeated multiplication of small numbers rather than by subtracting two nearby complex numbers.

We can continue this procedure through ρ_m, a cross-ratio involving w_p, w_q, and two prevertices of Q_m. Finally, we find the image of these points under the fractional

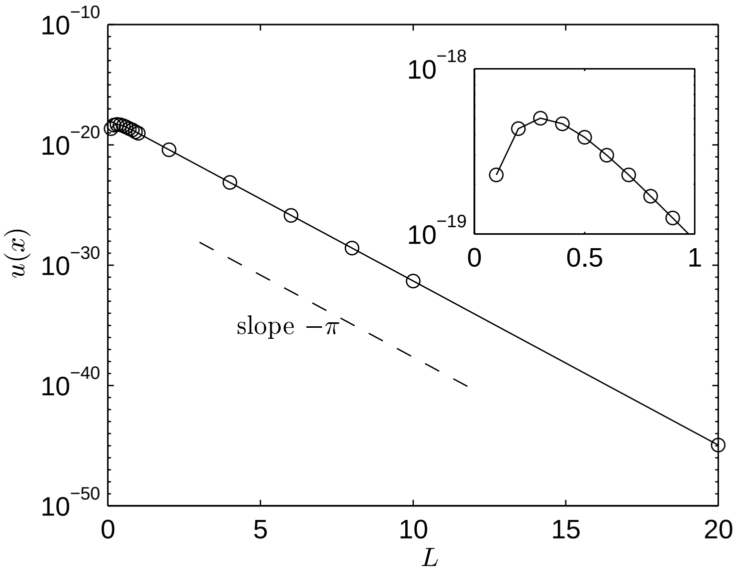

FIG. 9. *Solution of the BVP for the region in Fig.* 8. *The inset shows the solution at points for $L < 1$. As L grows, the curve quickly approaches $k \cdot e^{-\pi L}$, corresponding to the leading-order behavior of the solution for a rectangle.*

linear transformation that moves ξ to the origin. Since ξ is not close to the boundary of the disk, the well-separated points will remain uncrowded. The invariance of ρ_m under fractional linear transformation allows us to recover $|w_p - w_q|$, which is now the desired h.

In the situation where segment s is split by CRDT, we simply add up the contributions to $u(x)$ separately from each subsegment using the preceding algorithm. A Laplace solution is linear in the boundary data, and there cannot be any cancellation at this step because each contribution is positive.

We use the method described above to solve the BVP on the T-shaped region of Fig. 8. The region is parameterized by L, the length of the long arm of the T, and the forced edge s lies at the end of the shortest arm. The point x lies along the centerline of the long arm and at a distance 0.1 from the end.

In Fig. 9 we plot the solution $u(x)$ for L up to 20. After a transient phase (shown in the inset), the behavior very quickly approaches $k \cdot e^{-\pi L}$ for some constant k. This is easily replicated by replacing the T by a rectangle with forced edge at the top.

One issue that arises immediately is verification of the computed solutions. We do this by noting that the elongations not containing x or the forced edge are largely irrelevant to the solution. In Fig. 8, for example, we can shrink the right branch of the crossbar, decreasing the BVP solution by an exponentially small amount. Once this branch is sufficiently small, we can compute a rectangle map and solve the approximate BVP as described above. We have done this for several values of L and verified the CRDT solutions to at least 10 digits. The advantage of the CRDT method is that the solution is found for the original region directly without introducing any approximations. Furthermore, the CRDT method is fully automatic. In several experiments, we were always able to accurately solve this class of exponentially small harmonic-measure conformal mapping problems using the approximation process. But convergence was attained only with prohibitively laborious hand-tuning that would not be feasible for novice users of the Toolbox.

The key idea in the applications of this section is the quadrilateral graph. By always operating in steps between overlapping quadrilaterals, we ensure that each step is well conditioned, even though the eventual results of compounding the steps may vary over many orders of magnitude. We believe that this technique is central to the application of CRDT to potential theory problems.

10. Conclusions.

We have introduced CRDT, a new algorithm for finding the S–C prevertices on the unit circle for arbitrary bounded polygonal regions. The classical crowding problem is avoided through conformally equivalent re-embeddings of the prevertices so that the numerical mapping is always locally accurate. In addition, the nonlinear system chosen for numerical solution apparently has a monotonicity property that makes it easier to solve numerically than previously posed systems for the S–C parameter problem, although we cannot verify that this is so for all polygons. We have so far been unable to prove that CRDT always converges.

The polygon is first split so that long, narrow regions can be represented piecewise by well-conditioned triangles. A Delaunay triangulation of the resulting n-vertex polygon is computed and used to define $n-3$ quadrilaterals, whose diagonals appear as internal sides in the triangulation. The primitive variables of the nonlinear system are logarithms of the cross-ratios of the prevertices of those $n-3$ quadrilaterals. These cross-ratios define an infinite set of conformally equivalent configurations of the prevertices, each of which produces an S–C map to the same image polygon. The imposed constraints are on the magnitudes of the cross-ratios of the quadrilaterals in the image polygon.

The CRDT algorithm generally compares favorably with the SC Toolbox for MATLAB in numerical experiments. The principal exceptions are those regions which require a great many extra vertices to be added in the splitting phase of the algorithm. We do not know if there is a more effective splitting procedure. On the other hand, CRDT has no problem finding the prevertices for arbitrarily elongated polygons, something which no previous algorithm can claim.

We demonstrate the use of CRDT in applications. Figure 7 shows the rectangle map to a multiply elongated polygon for which we believe no other algorithm would work. We also illustrate how to use CRDT to solve a particular elliptic boundary value problem. In each case, the key is to follow a path in the quadrilateral graph of the polygon (dual of the triangulation). The distance separating prevertices can be computed without cancellation, even when the distance is extraordinarily small. This technique can be used to solve a certain Dirichlet problem or compute a conformal modulus to an accuracy not achievable by other methods on most regions.

Besides the unresolved matters already introduced in this work, there are open questions about possible extensions of CRDT to work with polygons with infinite vertices, circular-arc polygons [10], or multiply connected regions [5, 13]. Future work will include the application of CRDT to grid generation and the incorporation of CRDT into the SC Toolbox.

Acknowledgments. We would like to thank Nick Trefethen at Cornell, Marshall Bern and John Gilbert at Xerox, and David Eppstein at the University of California at Irvine for their suggestions and comments. We are indebted to Jack Snoeyink at the University of British Columbia for his proof of Proposition 1.

Note added in proof. We have incorrectly referred to local minima of nonlinear equations arising in the SC Toolbox. The situation that may arise is that the only descent direction leads to infinity.

REFERENCES

[1] B. S. Baker, E. Grosse, and C. S. Rafferty, *Nonobtuse triangulation of polygons*, Discrete Comput. Geom., 3 (1988), pp. 147–168.

[2] M. Bern and D. Eppstein, *Mesh generation and optimal triangulation*, in Computing in Euclidean Geometry, D. Z. Du and F. Hwang, eds., World Scientific, Singapore, 1992. Also Xerox Palo Alto Research Center Technical Report CSL-92-1, Palo Alto, CA.

[3] M. Bern, D. Eppstein, and J. Gilbert, *Provably good mesh generation*, in Proc. 31st IEEE Symp. on Foundations of Computer Science, 1990, pp. 231–241.

[4] L. P. Chew, *Guaranteed-Quality Triangular Meshes*, Technical Report 89–983, Department of Computer Science, Cornell University, Ithaca, NY, 1989.

[5] H. D. Däppen, *Die Schwarz–Christoffel-Abbildung für Zweifach Zusammenhängende Gebiete mit Anwendungen*, Ph.D. thesis, ETH Zürich, Switzerland, 1988.

[6] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[7] T. A. Driscoll, *A Matlab toolbox for Schwarz–Christoffel mapping*, ACM Trans. Math. Software, 22 (1996), pp. 168–186.

[8] P. Henrici, *Applied and Computational Complex Analysis*, Vol. 1, Wiley, New York, 1974.

[9] L. H. Howell, *Computation of Conformal Maps by Modified Schwarz–Christoffel Transformations*, Ph.D. thesis, MIT, Cambridge, MA, 1990.

[10] L. H. Howell, *Numerical conformal mapping of circular arc polygons*, J. Comput. Appl. Math., 46 (1993), pp. 7–28.

[11] L. H. Howell, *Schwarz–Christoffel methods for multiply-elongated regions*, in Proc. 14th IMACS World Congress on Computation and Applied Mathematics, 1994.

[12] L. H. Howell and L. N. Trefethen, *A modified Schwarz–Christoffel transformation for elongated regions*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 928–949.

[13] C. Hu, *User's guide to DSCPACK*, Nat. Inst. Aviation Res. 95-1, Wichita State Univ., Wichita, KS, 1995.

[14] R. Menikoff and C. Zemach, *Methods for numerical conformal mapping*, J. Comput. Phys., 36 (1980), pp. 366–410.

[15] S. A. Mitchell and S. A. Vavasis, *Quality mesh generation in three dimensions*, in Proc. 8th ACM Symp. on Computational Geometry, 1992, pp. 212–221.

[16] Z. Nehari, *Conformal Mapping*, Dover, New York, 1952.

[17] J. Snoeyink, *Cross ratios and angles determine a polygon*, Personal communication, August 1996.

[18] L. N. Trefethen, *Numerical computation of the Schwarz-Christoffel transformation*, SIAM J. Sci. Statist. Comput., 1 (1980), pp. 82–102.

[19] L. N. Trefethen, *Numerical construction of conformal maps*, Appendix to Fundamentals of Complex Analysis for Mathematics, Science, and Engineering, by E. B. Saff and A. D. Snider, Prentice-Hall, Englewood Cliffs, NJ, 1993.