

# Algorithm 843: Improvements to the Schwarz–Christoffel Toolbox for MATLAB

TOBIN A. DRISCOLL  
University of Delaware

---

The Schwarz–Christoffel Toolbox (SC Toolbox) for MATLAB, first released in 1994, made possible the interactive creation and visualization of conformal maps to regions bounded by polygons. The most recent release supports new features, including an object-oriented command-line interface model, new algorithms for multiply elongated and multiple-sheeted regions, and a module for solving Laplace’s equation on a polygon with Dirichlet and homogeneous Neumann conditions. Brief examples are given to demonstrate the new capabilities.

Categories and Subject Descriptors: G.1.8 [Numerical Analysis]: Partial Differential Equations—*Elliptic equations*; J.2 [Physical Sciences and Engineering]—*Mathematics and statistics, Engineering, Physics*; G.4 [Mathematical software]—*User interfaces*

General Terms: Algorithms

Additional Key Words and Phrases: MATLAB, conformal mapping, Laplace’s equation, polygons, Schwarz–Christoffel

---

## 1. INTRODUCTION

The Schwarz–Christoffel (SC) formula is a semi-explicit expression for a conformal map from a standard region to the interior of a polygon. For instance, the map from  $H$ , the upper half of the complex plane, can be written as [Henrici 1974]

$$f(z) = f(z_*) + c \int_{z_*}^z \prod_{k=1}^{n-1} (\zeta - x_k)^{\alpha_k - 1} d\zeta, \quad (1)$$

where  $z_*$  is a point in  $\bar{H}$  and the real, ordered *prevertices*  $x_1, \dots, x_{n-1}, \infty$  map in counterclockwise order to the vertices of the polygon whose interior angles are  $\pi\alpha_1, \dots, \pi\alpha_n$ . Vertices are permitted to lie at infinity, in which case their associated angles (interpreted on the Riemann sphere) are between  $-2\pi$  and 0. Three of the prevertices may be chosen arbitrarily, but for a target polygon

---

Supported by NSF grant DMS0104229.

Authors’ address: Department of Mathematical Sciences, University of Delaware, Ewing Hall, Newark, DE; email: (driscoll@math.udel.edu).

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 1515 Broadway, New York, NY 10036 USA, fax: +1 (212) 869-0481, or [permissions@acm.org](mailto:permissions@acm.org).

© 2005 ACM 0098-3500/05/0600-0239 \$5.00

with  $n > 3$  the rest must be determined from the geometry; this is called the *parameter problem*. Once this is solved, values of the map can be computed by quadrature. Many variations on (1) are possible, for example using a different standard region. For details on these, see Driscoll and Trefethen [2002].

The SC formula and its offspring were implemented computationally a number of times in the twentieth century [Haugeneder and Prochazka 1978; Davis 1979; Reppe 1979; Floryan 1985; Däppen 1987; Howell and Trefethen 1990; Hu 1998]. Of particular note was the Fortran SCPACK [Trefethen 1980], which was released into the public domain and featured fast and stable implementation of three key steps: numerical quadrature for (1), solution of the parameter problem, and numerical inversion of the map.

The Schwarz–Christoffel Toolbox (SC Toolbox) for MATLAB [Driscoll 1996], first released to the public domain in 1994, implemented and somewhat generalized the methods used in SCPACK. In addition to the natural exploitation of the interactive MATLAB command line, the features of the original SC Toolbox included:

- The ability to map to the interior of a polygon from a disk, half-plane, strip or rectangle, or to the exterior of a polygon from a disk;
- The computation of forward and inverse maps;
- The use of unbounded regions (polygons with infinite vertices);
- Automated visualization of maps;
- A complete graphical interface, including the interactive drawing of polygons.

The SC Toolbox has undergone continuing change and improvement since its initial release, gradually taking advantage of advances in algorithms, applications, and MATLAB itself. Compared to the original, the current stable release (2.3) includes additions that improve robustness, applicability, and convenience.

The SC Toolbox offers different fundamental domains and variants of the basic map (1) in large part because different domains are associated with instant solutions to certain problems—for example, strips are helpful in channel flows, rectangles are natural for studying resistors, and so on. However, one bad side effect in the original Toolbox was a proliferation of functions with similar names and parallel functionality, such as `dplot`, `deplot`, `hpplot`, `rplot`, and `stplot`. Furthermore, each type of map has a slightly different collection of values needed to define and work with it. The object-oriented (OO) capabilities of MATLAB, introduced since the original Toolbox release, allow significant improvements on these accounts. Function overloading, for instance, allows replacement of all the above commands with a single `plot` command that is conversant with the different variations. Data encapsulation shields the user from being concerned with the details of map parameters (although they are accessible if needed). Finally, inheritance allows one to abstract and reuse some features that all the different types of map share. Object orientation has also been used to aid in working with polygons, which are fundamental to SC maps. The OO aspects of the new Toolbox are described in Section 2.

The most important numerical issue in SC mapping is the *crowding phenomenon*, which is a form of ill conditioning common in computational

conformal mapping [Gaier 1972; Menikoff and Zemach 1980; Krikeles and Rubin 1988]. For domains that are “singly elongated,” like an L or Z shape, rectangle maps effectively deal with crowding. For a multiply elongated T or E shape, however, rectangles are not helpful. Instead, the CRDT algorithm [Driscoll and Vavasis 1998] allows stable computations for polygons with, in principle, arbitrary elongations, and it is now implemented by the Toolbox. Another extension of Toolbox functionality is to maps to Riemann surfaces—regions with branch points that cannot be embedded in a single sheet of the plane. The new functionalities are described in more detail in Section 3.

While different domains do make solutions of particular boundary-value problems easy, other problems require additional effort after conformal transplantation. In Driscoll and Trefethen [2002] a process for the fast direct solution of Laplace’s equation with piecewise constant Dirichlet and homogeneous Neumann data is given. The Toolbox now completely automates this process, including a GUI for assigning boundary values and viewing the results. These additions are described in Section 4.

Finally, in Section 5 we give a few short examples of code using the toolbox to illustrate the new features described here.

## 2. OBJECT-ORIENTED COMMAND LINE MODEL

Since a toolbox is intended to add functionality to MATLAB, in a sense the toolbox author is trying to collaborate with many future programmers. Thus it is quite natural to use an object-oriented (OO) model as an interface. Although MATLAB implements a rather limited OO model, the SC Toolbox benefits greatly from encapsulation, overloading, and (though to a lesser extent) inheritance.

Classes in MATLAB are defined by directories whose names start with an @ character. A class directory contains function M-files that are known as *methods*; these are (more or less) the only functions that operate on objects of this class. When a method name duplicates that of a built-in MATLAB function, the function is said to be *overloaded* and becomes extended to the class. A class may be allowed to *inherit* the methods of a parent class. Each class must have a *constructor method*, named identically to the class, that is responsible for creating objects of the class. The resulting object can be assigned to a variable in the MATLAB environment and may hold auxiliary data needed by the methods. This data is *encapsulated*, that is, not normally available to the user except through the interface provided by the methods.

Polygons play a central role in SC mapping and have a class definition in the SC Toolbox. A polygon object can be created by specifying it by a vector of complex vertices. For a polygon with finite vertices, the bounded region it encloses is considered its interior (which is detected automatically through angle sums). In SC mapping, polygons with vertices at infinity are sensible and allowed by the Toolbox; in this case one must also supply a vector of interior angles that defines the interior. One can also create polygons, including unbounded ones, graphically through the function `polyedit`. Simple transformations such as scaling, rotation, and translation are implemented by overloading the

Table I. Methods for the polygon Class

<code>polygon</code>	Construct a polygon object		
<code>vertex, double, ()</code>	Vertex extraction or assignment	<code>+, -, *</code>	Translation, scaling, rotation
<code>angle</code>	Angle extraction	<code>modify</code>	Graphically edit
<code>display</code>	Print out vertices	<code>truncate</code>	Truncate infinite sides
<code>plot, fill</code>	Show graphically	<code>linspace</code>	Equally spaced points on boundary
<code>size, length, isempty</code>	Number of vertices	<code>intersect</code>	Find intersections with a segment
<code>diam, isinf</code>	Diameter	<code>triangulate</code>	Triangulate interior
<code>winding, isinpoly</code>	Winding number	<code>cdt, plotcdt</code>	Constrained Delaunay triangulation of vertices

Table II. Schwarz–Christoffel Map Classes

<code>diskmap</code>	Map from the unit disk
<code>hplmap</code>	Map from the upper half-plane
<code>stripmap</code>	Map from a strip
<code>rectmap</code>	Map from a rectangle
<code>extermmap</code>	Map from the disk to polygon exterior
<code>crdiskmap, crrectmap</code>	Map using CRDT (Section 3)
<code>riesurfmap</code>	Map to a Riemann surface (Section 3)

MATLAB binary operators `*` and `+` to pair a polygon with a complex scalar. Other polygon methods are provided for convenience; a complete list is given in Table I. Mostly these are self-explanatory, but a few deserve special comment. Since SC maps to Riemann surfaces are possible (see Section 3), it is occasionally useful to use `winding` to find the winding number of the region with respect to a specific point. (For points well in the interior, the argument principle works well, but the computation for points very close to the boundary is unavoidably ill-conditioned.) The `truncate` method is present because while SC maps themselves have no difficulty with unbounded regions, such regions are inconvenient for certain graphical and computational purposes. Overloading of the built-in `linspace` provides a convenient way to discretize the boundary of a polygon, and `triangulate` produces a tunable triangulation of the interior of the domain<sup>1</sup> that is compatible with such functions as `trisurf`.

The SC Toolbox supports different map variations distinguished by fundamental domain, numerical algorithm, or other features. Each map type is a MATLAB class, as listed in Table II, that encapsulates the map parameters. In the typical case a map is constructed by supplying a polygon and perhaps additional data defining the map, although one can also create a map from specified prevertices and angles to an unknown geometry. The core method names associated with each map type are shown in Table III, although a few types lack some of these or add others. Note that many built-in MATLAB functions are

<sup>1</sup>However, the method used is not robust for slit regions—polygons having a vertex with interior angle  $2\pi$ .

Table III. Core Methods for the Classes in Table II

<i>classname</i>	Find parameters and construct map
<code>eval, feval</code>	Evaluate map at point(s)
<code>plot</code>	Plot an automated visualization
<code>evalinv</code>	Evaluate the inverse of the map
<code>evaldiff</code>	Evaluate the derivative of the map
<code>display</code>	Pretty-print
<code>parameters</code>	Extract map parameters
<code>accuracy</code>	Accuracy estimate

```

[w,beta] = drawpoly;                               p = polyedit;
plotpoly(w,beta)                                  plot(p)
[z,c,L,qdat] = rparam(w,beta);                     f = rectmap(p)
rdisp(w,beta,z,c,L)                               f(0.2i)
rmap(0.2i,w,beta,z,c,L,qdat)                       plot(f)
rplot(w,beta,z,c,L)

```

Fig. 1. Scripts of equivalent functionality using the original command-line interface (left) and the new object-oriented model (right).

overloaded to work with these maps. In addition, the map classes all descend from an `scmap` class that implements some tasks common to all map types, such as extraction of the polygon or computational options, or scaling and translation of the image. Two related classes, `scmapinv` and `scmapdiff`, allow the existence of abstract map inverses or derivatives as named objects that can then be evaluated. For the most part, the SC map classes implement functionality that was available prior to the support of OO programming in MATLAB. Thus most of the map methods are wrappers around older functions, which are still visible to the user in their original form. This leads to a slightly awkward structure but allows compatibility with legacy code.

An additional `moebius` class creates, pretty-prints, evaluates, and inverts Möbius (also called fractional linear) transformations. They may be created in terms of four explicit coefficients, or by specifying the images of three points. For example, we can use  $-i$ ,  $i$ , and  $\infty$  to define the imaginary axis, and  $0$ ,  $1 - i$ , and  $2$  to specify a circle of unit radius with center at  $z = 1$ . One mapping between these two generalized circles is therefore

```
moebius([-i i Inf],[0 1-i 2])
```

```
ans =
```

$$\frac{(2+2i) + (2-2i)*z}{(-1+3i) + (1-1i)*z}$$

As an example of the advantages offered to the interactive user by the OO model, consider the two scripts in Figure 1. Both scripts perform essentially the same core toolbox tasks of creating a polygon, finding a map to a rectangle, pretty-printing essential map details, evaluating the map at a point, and forming a visualization. However, the OO version displays considerably improved

conceptual clarity and also offers some concrete advantages:

- One variable name each is needed to refer to the polygon and to the map, whereas in the older model several variables are needed for each (and furthermore, different types of maps require different numbers of variables). This difference is even more significant when one wants to create multiple polygons and/or maps.
- Because of overloading, one can use familiar function names such as `plot` and `display`, the latter of which is automatically invoked when an object is the result of a statement not ending with a semicolon. In the original model, each type of map has its own family of functions with a different prefix.
- Thanks to overloading of `subsref`, the natural functional notation `f(0.2i)` can be used to indicate map evaluation.

An additional benefit of the OO model is a `composite` class that holds any number of individual maps and interprets evaluation, differentiation, and inversion accordingly. Such objects are useful, for example, to map one polygon to another using a standard region as an intermediary. A `composite` object can also contain `moebius` or `native inline` objects, though the latter type precludes differentiation and inversion. For a use of `composite` objects in this context, see Section 4.

Finally, it is expected that the OO model will allow considerable future expansion of the toolbox. For example, it would be useful to add algorithms for mapping to more general Jordan regions. In that case, `polygon` may become a descendant of a more general region class.

### 3. ALGORITHMIC IMPROVEMENTS

Two major additions have been made to the original toolbox's available algorithms. One of these is the CRDT algorithm [Driscoll and Vavasis 1998], useful for dealing with the ill-conditioning arising from crowding [Gaier 1972; Menikoff and Zemach 1980; Krikeles and Rubin 1988]. The canonical example of crowding is the map from the unit disk to a rectangle of aspect ratio  $a$ . In such a map, some pair of prevertices on the unit circle is separated by a distance that is an exponentially small function of  $a$ . Hence it becomes difficult to compute the map in the vicinity of such points accurately. A similar phenomenon results from any elongated part of a target region  $P$ . The CRDT algorithm overcomes crowding by efficiently representing the entire family of possible maps to  $P$ . Members of the family are linked by self-maps of the disk, and there is at least one family member that is well conditioned for maps to any particular small subdomain of  $P$ .

The primary disadvantage of CRDT is that it adds (potentially) many vertices to the straight sides of the polygon in order to keep local aspect ratios under control. Since the algorithm tends to be cubic in the number of vertices, CRDT can be relatively slow, and as a result it is not used in the standard `diskmap` class of the SC Toolbox. Instead it is offered in a separate class, `crdiskmap`, whose interface is essentially identical to that of `diskmap`. A related class, `crrectmap`, creates a "rectified" map as described in Driscoll and Vavasis [1998]. In this case

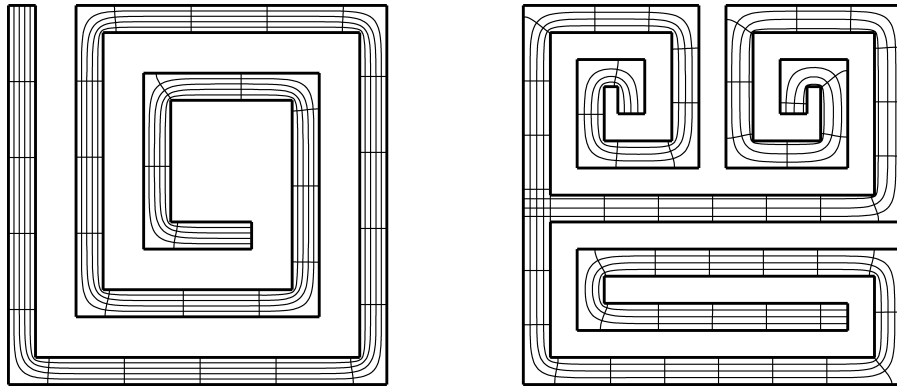


Fig. 2. Examples of the CRDT mapping algorithm. For regions on which other methods are viable (such as `rectmap` for the spiral at the left), CRDT is almost an order of magnitude slower. However, CRDT can be used for maps to regions with multiple elongations (right) for which no other known algorithm will work.

the disk is an intermediate step to another polygon whose boundary sides are parallel to the coordinate axes. Such a region is convenient for grid generation or finite differences.

As an example of what CRDT provides, consider Figure 2. Both regions shown are too elongated to be mapped using `diskmap`. The spiral on the left can be mapped to a rectangle using either `rectmap` or `crrectmap`, but the times needed on a typical PC are about 12.4 and 104 seconds, respectively.<sup>2</sup> The region on the right, however, has three global elongations and is most naturally mapped to a T-shaped region. Due to crowding, `crrectmap` is the only known SC mapping method that will work at all for this region.

The other algorithmic addition to the toolbox is the map to a Riemann surface [Gilbarg 1949]. Let  $P$  be a surface on  $s$  sheets bounded by a polygon. The map to  $P$  is a modification of (1):

$$f(z) = f(z_*) + c \int_{z_*}^z \prod_{b=1}^{s-1} (\zeta - z_b)(\zeta - \bar{z}_b) \prod_{k=1}^{n-1} (\zeta - x_k)^{\alpha_k - 1} d\zeta, \quad (2)$$

where  $z_1, \dots, z_{s-1}$  are the preimages of the branch points of  $P$ . Unlike other map classes, `riesurfmap` does not support an inverse map at this writing, because of the difficulty in finding all values of the multivalued inverse. Figure 3 shows the map to a two-sheeted region. These maps play a key role in the Laplace solver described in the next section.

#### 4. APPLICATION TO LAPLACE'S EQUATION

The most common and powerful application of conformal mapping is in the solution of Laplace's equation, which appears throughout engineering and physics [Henrici 1986; Costamagna 1987; Schinzinger and Laura 1991]. At a basic level, conformal mapping can be used to simplify the geometry and

<sup>2</sup>Pentium 4, 2.5-GHz, running MATLAB 6.5.1 under Linux 2.6.5.

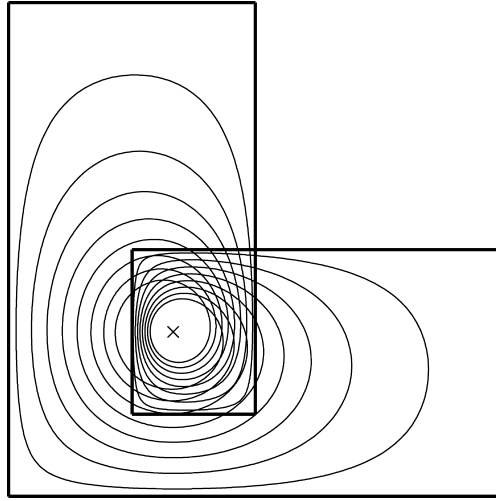


Fig. 3. Map to a two-sheeted Riemann surface. The branch point (marked by an 'x') is mapped to the origin in the disk, so that every circle around the origin maps to a two-loop curve in the region.

facilitate discretization. However, for a significant class of boundary values, an essentially explicit solution is possible. The following description is loosely based on Sections 5.1–5.3 of Driscoll and Trefethen [2002].

Let  $P$  be a domain with polygonal boundary  $\Gamma$ . Let  $\Gamma = \Gamma_1 \cup \dots \cup \Gamma_n$  be a finite disjoint partition in which  $\Gamma_k$  is connected and bounded by endpoints  $w_k$  and  $w_{k+1}$ . (Throughout this discussion we assume circular indexing, so for instance  $w_{n+1} = w_1$  and  $\Gamma_0 = \Gamma_n$ .) Each  $\Gamma_k$  may be thought of as a logical, but not necessarily geometric, side of  $\Gamma$ . Let  $K_D$  be a subset of indices from 1 to  $n$  and let  $K_N$  be its complement. We solve the boundary value problem

$$\Delta u = 0 \text{ in } P, \quad (3a)$$

$$u = u_k \text{ on } \Gamma_k, \quad k \in K_D, \quad (3b)$$

$$\frac{\partial u}{\partial n} = 0 \text{ on } \Gamma_k, \quad k \in K_N. \quad (3c)$$

The boundary conditions are piecewise constant with prescribed values (Dirichlet) or zero normal derivative (homogeneous Neumann). These are the conditions that are preserved by conformal mapping.<sup>3</sup> For the following discussion (but not for the coded algorithm) we assume that no two neighboring indices appear in  $K_N$  and that neighboring Dirichlet sides have different assigned values; in other words, we choose our logical sides as coarsely as possible.

The solution can be computed in two overall stages. The first stage is an ordinary SC map  $f(z)$  from the upper half-plane to the interior of  $P$ . The preimages  $x_1, \dots, x_{n-1}, \infty$  of the logical vertices  $w_1, \dots, w_n$  of  $\Gamma$  are determined by the solution of a geometrically derived set of nonlinear equations.

<sup>3</sup>In fact, a slightly more general form, in which the derivative is zero at a specified angle to  $\Gamma_k$ , is also preserved and admits a direct solution. See Driscoll and Trefethen [2002] for details. However, (3) is general enough for many applications.



In the second stage of the solution we further transform the upper half-plane via map  $g(z)$  to a special region  $R$ . This region is constructed so that under the composite map  $h = g \circ f^{-1}$  from  $P$  to  $R$ , each  $\Gamma_k$  for  $k \in K_D$  is transformed to a vertical side of  $R$ , and each  $\Gamma_k$  for  $k \in K_N$  is transformed to a horizontal side of  $R$ . There are infinitely many ways to do this, but when  $g$  is chosen so that  $\operatorname{Re} h$  is bounded and  $\operatorname{Re} h = u_k$  on  $\Gamma_k$  for  $k \in K_D$ , then it follows immediately that  $u = \operatorname{Re} h$  is the unique bounded solution to (3).

It is not hard to show that the second-stage transformation  $g$  is in fact another SC map. Let  $\kappa = |K_D|$ . If  $\kappa < 2$  the solution is constant; otherwise there exists a unique real polynomial  $p$  of degree no greater than  $\kappa - 2$  such that

$$g'(z) = cp(z) \prod_{k=1}^{n-1} (z - x_k)^{\beta_k}, \quad (4)$$

where  $\beta_k = -1$  if  $k - 1$  and  $k$  both belong to  $K_D$  and  $\beta_k = -1/2$  otherwise. The terms in the product simply ensure the appropriate transitions between logical sides, while  $p(z)$  introduces geometric flexibility to  $R$  needed to impose the Dirichlet values. Finally,  $c$  represents a global rotation:  $c = 1$  if  $n - 1 \in K_N$  and  $c = i$  otherwise.

Furthermore, the coefficients of  $p$  can be determined by the solution of a set of *linear* equations. By substituting each monomial  $1, \dots, z^{\kappa-2}$  for  $p$  in (4) and integrating between consecutive Dirichlet sides, one can create a square matrix  $M$ . Since the real part of  $g$  is prescribed on the Dirichlet sides, the coefficients are found by inverting  $\operatorname{Re} M$ . This process can be thought of as a generalization of the classical conformal modulus problem for quadrilaterals in the case  $\kappa = 2$  [Henrici 1986] and has appeared elsewhere in different guises [Widom 1969; Versnel 1983; Lax and Levermore 1983; Embree and Trefethen 1999].

Each real root of  $p$  creates a slit in the image  $R$ . By comparison to (2), we see that each complex conjugate pair of roots creates a branch point in  $R$ , which becomes multiple-sheeted. Note that this is possible only if  $\kappa > 3$ . This eventuality creates no serious computational obstacles and simply means that  $u$  assumes some values at more than one place in  $P$ .

In summary, finding the data needed to compute the solution to (3) requires solving the standard SC parameter problem of size  $n$ , followed by a linear system solution that takes an insignificant amount of time by comparison. Once the data have been found, computation of  $u$  at a particular point of  $P$  requires one SC formula inversion and one forward SC evaluation. For many regions, this process is very fast.

Compared to other methods that offer geometric flexibility, such as finite elements, the chief algorithmic advantages of the SC approach are: efficiency at high accuracy, since the errors in SC parameters can be reduced to near machine precision at very little cost once a few digits are known; effortless use in unbounded regions; and the incorporation of the exact nature of singularities that can significantly degrade the convergence of straightforward general-purpose methods. In a broader sense, the SC method is attractive because it produces

```

p = polygon([1 0.6 1 1i -1 -1i -0.4i -1i]);
f = center( diskmap(p), 0 );
plot(f)
eval(f,[0 0.1])

ans =
    -0.0000 + 0.0000i    0.0377 - 0.0547i

eval( diff(f), 0)

ans =
    0.3896 - 0.5526i

```

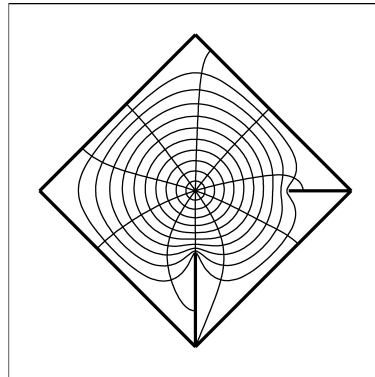


Fig. 4. Commands to construct and use a map from the disk to a polygon with two slits.

an exact solution, albeit one with a modest number of numerically determined parameters.

In the SC Toolbox, the user calls the function `lapsolve` with a polygon or map object and a vector of boundary data. The result is of class `composite` and can be evaluated at points in  $P$ . The `triangulate` method for polygons makes it easy to quickly visualize the solution using the built-in `trisurf` or `trimesh` functions. Usage is demonstrated in the next section.

## 5. EXAMPLES

Once its file archive has been downloaded and extracted, the SC Toolbox can be made available to MATLAB by adding its top-level directory to the MATLAB path (for example, using `addpath` or `pathtool` within MATLAB).

The toolbox is distributed with many sources of help: a user's guide in PDF format, online help accessible from the MATLAB command line using `help` or `doc`, and a group of tutorials that can be run by typing `scdemo`. Most of the functions of the toolbox can be accessed from the graphical interface that is started by typing `scgui`. In the rest of this section we give a few examples of using the SC Toolbox from the command line, with an emphasis on the additions described in this article.

*Basic maps.* Two examples of the basic process are shown in Figures 4 and 5. Whether the command-line or the graphical interface is used, the overall process is the same. First, a polygon is defined—that is, a polygon object is constructed. In these examples the vertices are specified numerically, but often one would instead draw them using `polyedit`, which can snap to a grid, discretize lengths and angles, and cope with infinite vertices. Second, the parameter problem is solved and the results are stored in a new map object. Finally, the map object can be evaluated, differentiated, inverted, or visualized.

*CRDT maps.* In this example (Figure 6) both `diskmap` and `rectmap` would fail due to multiple elongations in the target polygon. After `crdiskmap` succeeds,

```
p = polygon([-1+1i -1-1i 2-1i 2 0 1i]);
f = rectmap(p,[1 3 4 6]); % specify corners
plot(f,8,4)
rectangle(f)
```

```
ans =
    1.5708
    1.5708 +11.1798i
   -1.5708 +11.1798i
   -1.5708
```

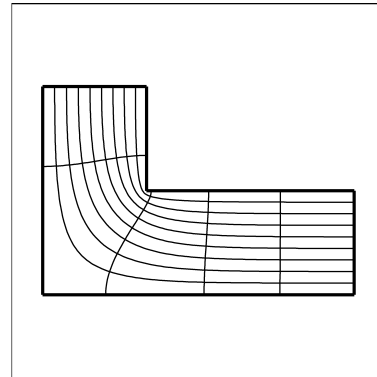


Fig. 5. Commands to construct and use a map from a rectangle.

```
arc = exp(i*pi*(0:10)/20);
p = polygon([0 fliplr(arc) 1.1*arc ...
            2i -0.1+2i -0.1]);
plot(p)
f = crdiskmap(p);
fr = crrctmap(f); % opens a GUI
plot( rectpoly(fr) )
```

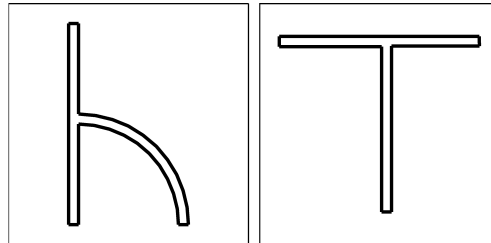


Fig. 6. Use of a CRDT map to rectify a polygon with multiple elongations.

```
p = polygon( exp(i*pi*(0:5)/3) );
u = lapsolve( p, [1 0 NaN 2 0 NaN] );
[tri,x,y] = triangulate(p,0.1);
ut = u(x+i*y);
trimesh(tri,x,y,ut)
[ class(u{1}), ' ', class(u{2})]
```

```
ans =
scmapinv , riesurfmap
```

```
u{3}
ans =
    Inline function:
    ans(z) = real(z)
```

```
plot(polygon(u{2}))
```

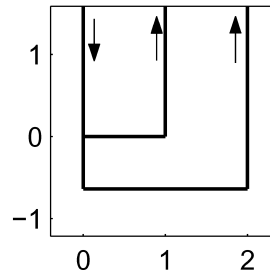
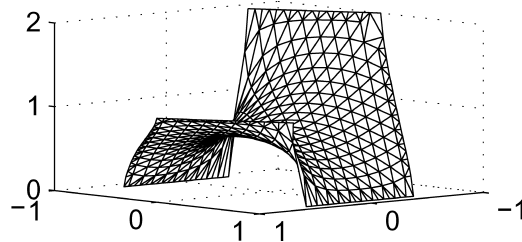


Fig. 7. Solution of Laplace’s equation on a hexagon. Shown are the solution and the derived conformally equivalent image whose real part leads to the solution (arrows added to designate sides going to infinity).

one can find a conformally equivalent rectified region with the aid of a graphical interface (not shown).

*Laplace's equation.* Figure 7 shows a solution of Laplace's equation on a regular hexagon, with two homogeneous Neumann sides and  $\kappa = 4$  Dirichlet sides. The boundary conditions can also be specified interactively using a graphical interface (not shown) that opens when they are not supplied in the call to `lapsolve`. The result of `lapsolve` is a composite object consisting of the inverse of an SC map, a map to a Riemann surface, and extraction of the real part. The image of the Riemann surface map is the region  $R$  from Section 4; in this example it has two sheets: the upper left corner is covered twice. On a 2.5-GHz PC (see footnote on p. 245), finding the composite map took less than half a second and evaluating at the 407 points in the triangulation took about 0.7 seconds. Internal consistency checks (not shown) automatically performed by the two SC maps involved indicate that the results should be accurate to about seven digits.

#### ACKNOWLEDGMENTS

I would like to acknowledge the continuing technical and moral support of Nick Trefethen, without whom the toolbox would not have been started nor continued. I also thank the numerous friends and colleagues who have shared careful bug reports, suggestions, applications, and kind words regarding the toolbox.

#### REFERENCES

- COSTAMAGNA, E. 1987. On the numerical inversion of the Schwarz–Christoffel conformal transformation. *IEEE Trans. Microw. Theory Tech.* 35, 35–40.
- DÄPPEN, H. 1987. Wind-tunnel wall corrections on a two-dimensional plate by conformal mapping. *AIAA J.* 25, 1527–1530.
- DAVIS, R. T. 1979. Numerical methods for coordinate generation based on Schwarz–Christoffel transformations. In *4th AIAA Computational Fluid Dynamics Conference*. Williamsburg, VA, 1–15.
- DRISCOLL, T. A. 1996. Algorithm 756: A MATLAB toolbox for Schwarz–Christoffel mapping. *ACM Trans. Math. Softw.* 22, 168–186.
- DRISCOLL, T. A. AND TREFETHEN, L. N. 2002. *Schwarz–Christoffel Mapping*. Cambridge University Press, Cambridge, UK.
- DRISCOLL, T. A. AND VAVASIS, S. A. 1998. Numerical conformal mapping using cross-ratios and Delaunay triangulation. *SIAM J. Sci. Comput.* 19, 1783–1803.
- EMBREE, M. AND TREFETHEN, L. N. 1999. Green's functions for multiply connected domains via conformal mapping. *SIAM Review* 41, 745–761.
- FLORYAN, J. M. 1985. Conformal-mapping-based coordinate generation method for channel flows. *J. Comput. Phys.* 58, 229–245.
- GAIER, D. 1972. Ermittlung des konformen Moduls von Vierecken mit Differenzenmethoden. *Numer. Math.* 19, 179–194.
- GILBARG, D. 1949. A generalization of the Schwarz–Christoffel transformation. *Proc. Nat. Acad. Sci.* 35, 609–612.
- HAUGENEDER, E. AND PROCHAZKA, W. 1978. Automatische Berechnung der Durchbiegungen und der Schnittgrößen dünner Platten mit Hilfe der Funktionentheorie. *Bauingenieur* 53, 243–248.
- HENRICI, P. 1974. *Applied and Computational Complex Analysis, Volume 1: Power Series, Integration, Conformal Mapping, Location of Zeros*. Wiley.
- HENRICI, P. 1986. *Applied and Computational Complex Analysis, Volume 3: Discrete Fourier Analysis, Cauchy Integrals, Construction of Conformal Maps, Univalent Functions*. Wiley.

- HOWELL, L. H. AND TREFETHEN, L. N. 1990. A modified Schwarz–Christoffel transformation for elongated regions. *SIAM J. Sci. Stat. Comput.* 11, 928–949.
- HU, C. 1998. Algorithm 785: A software package for computing Schwarz–Christoffel conformal transformation for doubly connected polygonal regions. *ACM Trans. Math. Soft.* 24, 3, 317–333.
- KRIKELES, B. C. AND RUBIN, R. L. 1988. On the crowding of parameters associated with Schwarz–Christoffel transformations. *Appl. Math. Comp.* 28, 297–308.
- LAX, P. D. AND LEVERMORE, C. D. 1983. The small dispersion limit of the Korteweg–de Vries equation II. *Comm. Pure Appl. Math.* 36, 571–593.
- MENIKOFF, R. AND ZEMACH, C. 1980. Methods for numerical conformal mapping. *J. Comput. Phys.* 36, 366–410.
- REPPE, K. 1979. Berechnung von Magnetfeldern mit Hilfe der konformen Abbildung durch numerische Integration der Abbildungsfunktion von Schwarz–Christoffel. *Siemens Forsch. u. Entwickl. Ber.* 8, 190–195.
- SCHINZINGER, R. AND LAURA, P. A. A. 1991. *Conformal Mapping: Methods and Applications*. Elsevier, Amsterdam.
- TREFETHEN, L. N. 1980. Numerical computation of the Schwarz–Christoffel transformation. *SIAM J. Sci. Stat. Comput.* 1, 82–102.
- VERSNEL, W. 1983. Electrical characteristics of an anisotropic semiconductor sample of circular shape with finite contacts. *J. Appl. Phys.* 54, 916–921.
- WIDOM, H. 1969. Extremal polynomials associated with a system of curves in the complex plane. *Advances in Math.* 3, 127–232.

Received January 2003; revised January and December 2004; accepted January 2005