An improved Schwarz-Christoffel Toolbox for MATLAB

Tobin A. Driscoll
Department of Mathematical Sciences
University of Delaware
Newark, DE 19716

Technical Report No. 2003-03

DEPARTMENT
OF
MATHEMATICAL SCIENCES

University of Delaware
Newark, Delaware

An improved 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 [**Computer Applications**]: Physical Sciences and Engineering—*mathematics and statistics, engineering, physics*; G.4 [**Mathematical software**]: User Interfaces—MATLAB

General Terms: Algorithms

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

1. INTRODUCTION

The Schwarz–Christoffel (SC) formula is a semi-explicit 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]

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

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

Department of Mathematical Sciences, Ewing Hall, Newark, DE 19716 (driscoll@math.udel.edu).
Supported by NSF grant DMS0104229.

the map.

The Schwarz–Christoffel Toolbox (SC Toolbox) for MATLAB [Driscoll 1996], first released to the public domain in 1994, implemented and generalized somewhat 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 from a disk, half-plane, strip or rectangle, or to the exterior of a polygon;

—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:

—A command-line object-oriented interface, with simplified calling sequences and overloading of familiar MATLAB functions;

—Convenience functions such as Möbius transformations, composite maps, and additional polygon functions;

—Use of the CRDT algorithm [Driscoll and Vavasis 1998] to allow stable computations for polygons with multiple elongations;

—Maps to Riemann surfaces;

—A function to solve Laplace's equation on a polygon with piecewise-constant Dirichlet and homogeneous Neumann conditions.

In section 2 we discuss how an object-oriented command interface model simplifies use of the Toolbox. In section 3 we briefly describe the algorithmic additions to the Toolbox; detailed presentation of the numerical methods can be found elsewhere. In section 4 we show how SC maps are used to solve Laplace's equation with high accuracy and speed. 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 (to a much 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*,

Table I. Methods for the `polygon` class.

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

Table II. Schwarz–Christoffel map classes.

`diskmap`	Map from the unit disk
`hplmap`	Map from the upper half-plane
`stripmap`	Map from a strip
`rectmap`	Map from a rectangle
`extermap`	Map from the disk to polygon exterior
`crdiskmap`, `crrectmap`	Map using CRDT (section 3)
`riesurfmap`	Map to a Riemann surface (section 3)

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 Toolbox. Most often a polygon object is specified by a vector of vertices and (optionally, if it is bounded) interior angles, or created graphically through the function `polyedit`. Simple transformations such as scaling, rotation, and translation are implemented by overloading the 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.

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 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 map classes implement functionality that was available prior to the support of OO programming in MATLAB. Thus most of the map meth-

Table III. Core methods for the classes in Table II.

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

ods 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 compatability with legacy code.

Finally, two other classes are occasionally convenient. The `moebius` class creates, evaluates, and inverts a Möbius (also called fractional linear) transformation either from its coefficients or by specifying the images of three points. The `composite` class acts as a container for a sequence of SC, Möbius, and MATLAB-native `inline` transformations.

3. ALGORITHMIC IMPROVEMENTS

Two major additions have been made to the toolbox's available algorithms. One of these is the CRDT algorithm [Driscoll and Vavasis 1998].

CRDT is useful because of 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]. 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.

Since CRDT tends to be more costly than the traditional mapping algorithm, 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 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.

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):

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

where $z_1, \ldots, z_{s-1}$ 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. The primary utility of this map type is in support of the Laplace solver described in section 4.

4. SOLVING 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 facilitate discretization. However, for a very important class of boundary values, much more 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 Γ. Let $\Gamma = \Gamma_1 \cup \cdots \cup \Gamma_n$ be a finite disjoint partition in which Γ_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 Γ_k may be thought of as a logical, but not necessarily geometric, side of Γ. 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

$$\Delta u = 0 \text{ in } P, \tag{3a}$$

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

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

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.[1] 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 $x_1, \ldots, x_{n-1}, \infty$ of the logical vertices $w_1, \ldots, w_n$ of Γ are determined by the solution of a geometrically derived set of nonlinear equations.

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 Γ_k for $k \in K_D$ is transformed to a vertical side of R, and each Γ_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 Γ_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

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

unique real polynomial p of degree no greater than $\kappa - 2$ such that

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

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 $1, \ldots, z^{\kappa-2}$ 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 before 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.

In the 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.

5. USAGE 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 used from the graphical interface that is started by typing `scgui`. In the rest of this section we give a few examples of using the Toolbox from the command line, with an emphasis on the additions described in this paper.

Basic maps. Two examples of the basic process are shown in Figures 1 and 2. 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

```
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
```

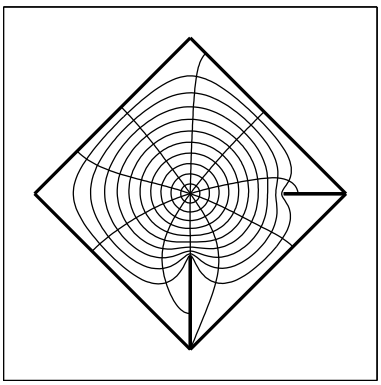

Fig. 1. Commands to construct and use a map from the disk, and the resulting visualization.

```
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
```

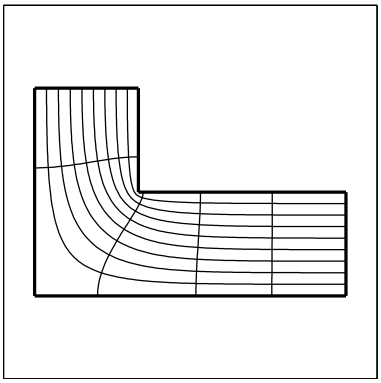

Fig. 2. Commands to construct and use a map from a rectangle.

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 3) both `diskmap` and `rectmap` would fail due to multiple elongations in the target polygon. After `crdiskmap` succeeds, one can find a conformally equivalent rectified region with the aid of a graphical interface (not shown).

Laplace's equation. Laplace's equation is solved on a regular hexagon, with two homogeneous Neumann sides and $\kappa = 4$ Dirichlet sides. (Figure 4). 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 an 800-MHz PC, finding the composite map

```
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 = crrectmap(f);  % opens a GUI
plot( rectpoly(fr) )
```

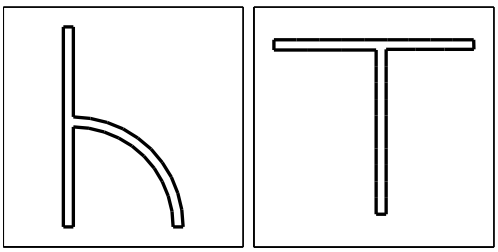

Fig. 3. Use of a CRDT map to rectify a polygon with multiple elongations.

```
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}))
```

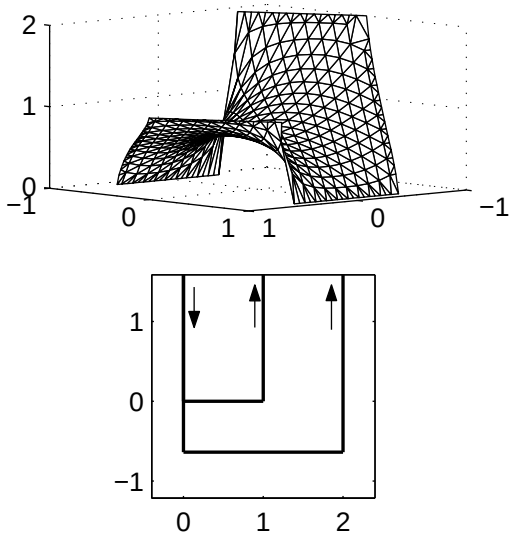

Fig. 4. Solution of Laplace's equation on a hexagon. Shown are the solution and the derived conformally equivalent image whose real part leads to the solution (arrows added to designate sides going to infinity).

took about half a second and evaluating at the 407 points in the triangulation took about 1.5 seconds. The results should be accurate to around six digits.

ACKNOWLEDGMENT

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 Comput. Fluid Dynamics Conf.* Williamsburg, VA, 1–15.

Driscoll, T. A. 1996. Algorithm 756: A MATLAB toolbox for Schwarz–Christoffel mapping. *ACM Trans. Math. Soft. 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.

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.