Schwarz-Christoffel Toolbox
User's Guide

Tobin A. Driscoll*

TR 94-1422
May 1994

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

* Center for Applied Mathematics, Cornell University, Ithaca, NY 14853
(driscoll@cam.cornell.edu).

Schwarz-Christoffel Toolbox User's Guide

Version 1.1

Tobin A. Driscoll*

Introduction

The Schwarz-Christoffel Toolbox (SC Toolbox) is a collection of files for the interactive computation and visualization of Schwarz-Christoffel conformal maps in MATLAB[1] version 4. The toolbox is a descendant of SCPACK, a Fortran package developed by L. N. Trefethen in the early 1980's [9]. However, the SC Toolbox has been written entirely in the MATLAB language, requires no programming by the user, and has many capabilities not in SCPACK.

The problem

The Schwarz-Christoffel formula is a recipe for a conformal map f from the upper half-plane to the interior of a polygon in the complex plane. The "polygon" may have slits or vertices at infinity, and can be characterized by two n-vectors: w, which contains the vertices in positively-oriented order, and β, which describes the corresponding *turning angles*. The pre-images of the vertices, or *prevertices*, are real and denoted by the vector x. They satisfy

$$x_1 < x_2 < \cdots < x_n = \infty.$$

Figure 1 illustrates these definitions.

If vertex w_j is finite, $\beta_j \pi$ is the exterior turning angle of the polygon at w_j, with a minus sign for left (counterclockwise) turns. If w_j is infinite, $\beta_j \pi$ is equal to -2π plus the exterior angle at the intersection of the two semi-infinite polygon sides as they are extended away from infinity. This is consistent with the interpretation of "turn." See Figure 1 for an example. Note that if w_j is finite, then $-1 < \beta_j \le 1$, with equality at the end of a slit, and if $w_j = \infty$, then $-3 \le \beta_j \le -1$. Furthermore, the positive orientation implies that if the target region is the polygon's interior, then necessarily

$$\sum_{j=1}^{n} \beta_j = -2.$$

*Center for Applied Mathematics, Cornell University, Ithaca, NY 14853 (driscoll@cam.cornell.edu).

[1]MATLAB is a registered trademark of The MathWorks, Inc.

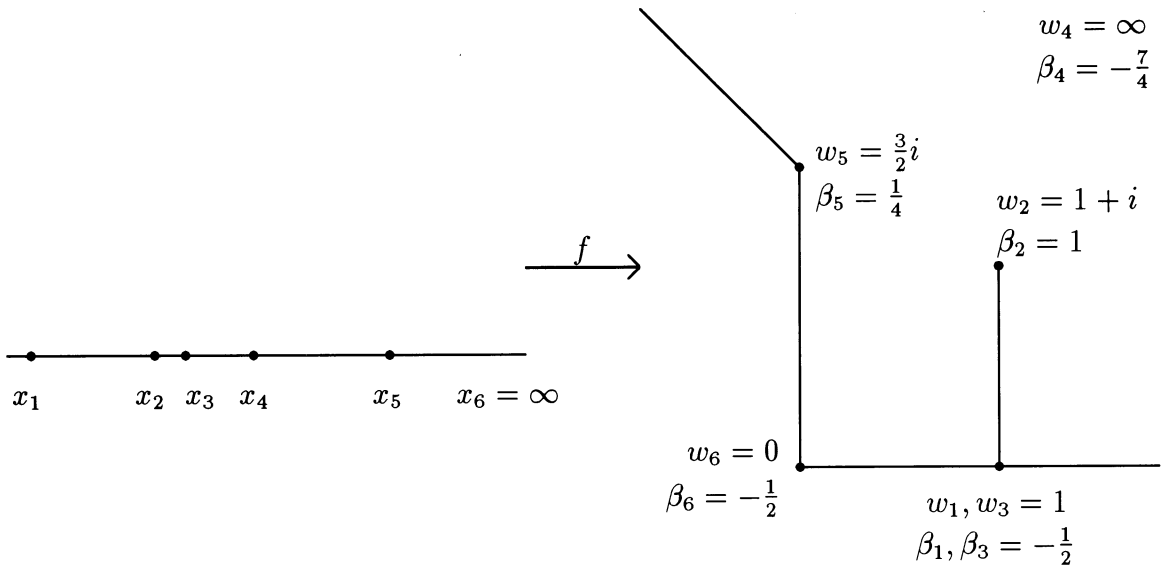


Figure 1: An example with $n = 6$.

The *Schwarz-Christoffel formula* for f is

$$f(z) = f(z_0) + c \int_{z_0}^{z} \prod_{j=1}^{n-1} (s - x_j)^{\beta_j} \, ds.$$

The main difficulty with this formula is that except in special cases, the prevertices x_j cannot be computed analytically. Three of them, including the already fixed x_n, may be be chosen arbitrarily (by the Riemann Mapping Theorem). The remaining $n - 3$ are then determined uniquely and can be obtained by solving a system of nonlinear equations. This is known as the *Schwarz-Christoffel parameter problem*, and its numerical solution requires careful treatment of the integration in the formula. Once the parameter problem is solved, the multiplicative constant c can be found and f and its inverse can be computed numerically.

The Schwarz-Christoffel map can be modified to fit some other situations. For example, if the fundamental domain is the unit disk rather than the upper half-plane, the prevertices z_j lie counterclockwise on the unit circle and the resulting formula is identical, except that the product has n terms rather than $n - 1$. Another instance is the exterior map, in which the fundamental domain is the unit disk and the target region is the exterior of a polygon. In this case the integrand has an additional singularity in the interior of the disk. If this singularity is fixed at the origin, only one prevertex may be chosen arbitrarily.

Toolbox features

- Graphical input of polygons
- Solution of the parameter problem for half-plane, disk, and exterior mapping
- Computation of forward and inverse maps

- Adaptive plotting of images of orthogonal grids
- Command-line and graphical user interfaces

Requirements

Platform

The SC Toolbox was developed on color Sun SPARCstations running MATLAB version 4.1. It will not run under any previous MATLAB version on this architecture. I have also used the toolbox with MATLAB 4.0 for MS-Windows, and have encountered only minor differences, which are explained throughout this guide.

Nothing is known about the performance of the toolbox on other platforms. I expect that the toolbox will run with few problems, but you may contact me if you experience difficulties.

A mouse is highly recommended.

NESOLVE

The Schwarz-Christoffel parameter problem leads to a set of nonlinear equations to be solved. For this it uses the package NESOLVE, part of the NONLINPK package written by Richard Behrens. NESOLVE is in the public domain and is included in the distribution of the SC Toolbox.

The original version of the SC Toolbox used, if available, the function FSOLVE in the Optimization Toolbox available from MathWorks. However, the algorithm used by FSOLVE causes it to be much slower than NESOLVE for most problems arising in the SC Toolbox, so its use has been discontinued.

User

Full use of the SC Toolbox requires some understanding of MATLAB fundamentals such as vectors, matrices, and functions. A working knowledge of how the `hold`, `axis`, and `plot` commands work in version 4 is helpful when plotting results. The tutorial that is shipped with MATLAB should be more than sufficient.

The graphical user interface should be accessible to MATLAB novices and experts alike.

Obtaining and installing the SC Toolbox

The latest version of the SC Toolbox is available by anonymous ftp at `ftp.cs.cornell.edu` in the directory `pub/driscoll/SC-Toolbox`. There is a shar (`*.sh`) file as well as a Postscript copy of the latest version of this guide.

In Unix a shar file is unpacked by entering `sh` *filename*; the files `unshar.zip` and `unshar-15.hqx`, also at the ftp site, are for unpacking the shar file on DOS and Macintosh systems, respectively. Under Unix, the toolbox is automatically unpacked into a subdirectory `sc` of the current directory. You may rename this subdirectory and put it wherever

you like—the most natural place is in your personal MATLAB directory. When you want to use the SC Toolbox, you must first use MATLAB's `cd` command to change to the toolbox subdirectory, or add it to the search path using `path`. Put the `path` command into your `startup.m` file if you want to have access to the toolbox by default in every MATLAB session.

As was mentioned in the Introduction, the SC Toolbox uses the package NESOLVE from Richard Behrens' NONLINPK. The necessary files are automatically unpacked into the SC Toolbox directory. If you already have a copy of NESOLVE on your MATLAB path, you may wish to delete all the files whose names start with `ne` in the SC Toolbox directory.

All the files listed above are also available at MathWorks' ftp site, `ftp.mathworks.com`. The toolbox and Postscript files are in the directory `pub/contrib/misc`, and shar file utilities can be found in `pub/contrib/tools`. A copy of NONLINPK is in `pub/contrib/optim`.

Using the SC Toolbox

The Schwarz-Christoffel Toolbox currently implements three variations of the Schwarz-Christoffel map: half plane to a polygon, disk to a polygon, and disk to the exterior of a polygon. Each variation has a module of functions given the prefixes `hp`, `d`, and `de`, respectively. The major functions comprising a module have the following suffixes:

`param`	Solves the parameter problem (supplies data needed by the other routines).
`disp`	Displays the results in a convenient format.
`map`	Computes the forward map.
`invmap`	Computes the inverse map.
`plot`	Plots the image of an orthogonal grid.

For example, one would use `hpparam` to solve the half-plane parameter problem and `hpplot` to get a quick visualization of the result. The functions are designed to have identical, or at least parallel, calling sequences and return values. In addition, each module has several utility files with the same prefix. These are not typically called interactively. Routines which have a prefix `sc` are general-purpose and are used by all the modules.

The function `drawpoly` provides a convenient way to input polygons, using a mouse. Alternatively, `scangle` can be used to compute turning angles at finite vertices. `plotpoly` can be used to plot a polygon given its vertices and turning angles.

Some additional routines are described at the end of this section. To see a menu of online demonstrations, type `scdemo` at the prompt.

Many routines take one or more optional arguments, which appear inside braces in the argument lists. An empty matrix `[]` supplied as an optional argument is equivalent to omission of that argument. This lets you supply a value for just the third optional argument, for example.

Graphical input of polygons

```
[w,beta] = drawpoly{(fig,axlim)}
```

Once `drawpoly` has been invoked, use the mouse to move the pointer into the figure window. The pointer will become a crosshair. Move to the place the first vertex will go and click the left mouse button. The vertex will be drawn, and you can move the mouse to place the next vertex. As you move, a dashed line will follow to preview the next polygon edge. Click the left button again, and the vertex and a solid polygon edge will be drawn. Continue in this manner until you wish to place the last vertex. For this you must use a different mouse button, or double-click. Then the polygon will be closed and the function will terminate.

The returned vector `w` contains the polygon vertices, as complex numbers, in the order they were entered. The vector `beta` gives the turning angles. If a proper polygon was entered in counterclockwise order, then `sum(beta)` should equal -2 (up to rounding errors).

If integer `fig` is supplied, it is the figure used for drawing; otherwise the current figure is used. By default the axes limits are reset to predetermined values, but if a vector `axlim` is given, it is used for the axes limits.

Note to MS-Windows users: To use `drawpoly`, you must first select **Enable background process** from the **Options** menu of the command window. This must be done each time you start MATLAB. Also, the preview line works differently: it appears only when you press and hold the mouse button, and will follow as you then drag the mouse. Release the button to place the vertex.

Restricting to a grid

You can easily place some or all of your vertices exactly on a discrete grid. This feature is controlled by the checkbox[2] labeled `Restrict to grid`. When this box is activated, a grid of dotted lines is drawn to indicate the allowable vertex sites. By default, the grid is spaced by 1/16 of the axes dimensions in each direction; you can vary that fraction between 1/4 and 1/32 with the slider[3] underneath the checkbox. As you move the pointer, the preview line will now end only on a grid point. You may continue to place vertices as before. When the checkbox is deactivated, the grid lines will be erased and the restriction removed.

Quantizing lengths and angles

You can also restrict some or all of the side lengths and turning angles of the polygon to be multiples of a fixed amount. This can be useful for drawing parts of regular polygons. Two checkboxes in the figure window control these modes.

When side lengths are quantized, the preview line will end only at points which give the resulting side a length that is an integer multiple of a fraction of the axes width. The initial value of that fraction is 1/8 and can be varied between 1/3 and 1/20 with the slider underneath the checkbox. When angles are quantized, the angle between the preview line and the most recent side is restricted to integer multiples of π/m, where m is initially 12

[2] A checkbox toggles between two states each time you click the mouse on it.

[3] A slider's value can be changed by pressing and holding a mouse button on the slider's bar, moving the mouse left or right, and releasing the mouse button. Ordinarily a slider takes on continuous values, but in `drawpoly` all values are rounded to integers.

and is slider-adjustable from 2 to 24. The angle quantization feature is particularly handy for drawing slits.

The quantization modes can be used independently or simultaneously. The quantization and grid modes are mutually exclusive, and activating one automatically deactivates the other(s).

Infinite vertices

To put a vertex at infinity, click the mouse outside the axes box at the proper exit angle. The mouse pointer will change to a cross. Now click again outside the axes box where you want the edge from infinity to return. (No preview line will follow between these clicks, since no edge actually exists.) The pointer will then revert to a crosshair.

Now move to place the next (finite, please!) vertex. If you move the mouse to a point such that the resulting edge would intersect the previous edge at a *finite* point outside the axes box, the preview line represents the projection of the current point onto the edge returning from infinity with a turn of -1. Also, the length-quantization mode will be temporarily disabled, but the other special modes will work as expected.

Solving the parameter problem

`[z,c,qdat] = xxparam(w,beta{,z0,options})`

(Again, braces indicate optional arguments.) The input parameters `w` and `beta` describe the polygon. The optional argument `z0` is for supplying an initial guess for the prevertices, and the meaning of the `options` vector is available in `scparmopt`. The output parameters are the prevertices `z`, the multiplicative constant `c`, and `qdat`, which is a matrix of quadrature data that the other routines may require.

Note: The first, second, and next-to-last vertices must be finite. If this is a problem, try renumbering or adding trivial (zero-turn) vertices.

Forward map

`xxmap(zp,w,beta,z,c{,qdat|tol})`

(The `|` represents "or.") The points to be mapped are in vector `zp`; the other parameters have been described above. The optional argument may either be `qdat` as described above or a scalar tolerance for the desired accuracy. If the mapping function is to be called many times, `qdat` should be supplied rather than recomputed automatically for each call.

Inverse map

`xxinvmap(wp,w,beta,z,c{,qdat,z0,options})`

The points whose pre-images are to be found are in `wp`. As with the forward map, the argument `qdat` is not required but is recommended if many calls are made. The optional

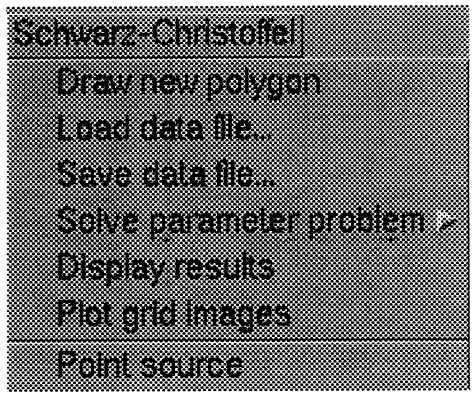


Figure 2: Main GUI menu.

parameter `z0` allows manual override of the starting points of the computations and is ordinarily unused. The meaning of `options` is documented in `scimapopt`.

Graphical output

xx`plot(w,beta,z,c{,var1|m,var2|n,options})`

Each of these functions plots the image of a standard orthogonal grid in the fundamental domain. The grid is cartesian for the half-plane and polar for the disk. With none of the optional arguments, ten evenly spaced curves in each direction are used. If integers `m` and `n` are given, the appropriate numbers of curves in each direction are mapped. With vectors `var1` and `var2` specified, the real and imaginary parts, or radii and arguments, of the curves in the grid are supplied by the user. The `options` argument is described in `scplotopt`.

The resulting curves should be smooth and intersect at right angles. However, adaptive refinement is done based on the region inside the axes box at the time of the call (or the region resulting from autoscaling), so if particular regions are of interest, set the axes limits via `axis` and issue a `hold on` command *before* invoking *xx*`plot`.

Graphical user interface (GUI)

The graphical user interface is designed to let you use all the major functions of the modules without using the command line.

To start using the GUI, create the menu system by invoking `scgui`. The current figure, or a new one if none exists, will be cleared and given a new menu called **Schwarz-Christoffel**. To see the available menu items, press and hold the left mouse button on this text. On a Sun workstation, the menu should look something like Figure 2. The items on the main part of this menu (above the line) are the general-purpose actions available. They are:

Draw new polygon	Clear the figure and input a polygon using `drawpoly`.
Load/Save data file...	Use a file with variables `w`, `beta`, `z`, `c`, `maptype`.
Solve parameter problem	Solve a particular variation of the parameter problem.
Display results	Invoke *xx*`disp`.
Plot grid images	Invoke *xx*`plot` with default curves.

To select an item, drag the mouse pointer to it and release the button. (The **Solve...** menu has a submenu to select from.) Many items are greyed out at first, indicating their functions are currently unavailable. When `scgui` is first called, only the first two are available, because the other operations are meaningless. Once a polygon has been input, the **Solve...** option becomes enabled, and once a solution has been computed, the other choices become active.

It is possible to modify and retrieve the polygon and solution data using `scgget` and `scgset`. These functions work somewhat like MATLAB's built-in `get` and `set`. The syntaxes are

```
{[value1,...]} = scgget(figh{,'property1'...})      or
        [w,beta,z,c,maptype] = scgget(figh)
```

and

```
scgset(figh,'property1',value1{,'property2',value2...})      or
                        scgset(figh,'clear')
```

In each case `figh` is a figure handle (equivalently, figure number). The valid properties are

`vertices`	Polygon vertices
`angles`	Polygon turning angles
`prevertices`	Obtained from `xxparam`
`constant`	Multiplicative constant of the map
`maptype`	One of the following:
	`'hp2p'` half-plane to polygon
	`'d2p'` disk to polygon
	`'d2ep'` disk to exterior polygon

Only the first three characters of the property name need be specified. The second form of `scgget` returns all of the available properties in the given order. The second form of `scgset` destroys all data relating to the current polygon.

While you may set properties by hand, the menus will not be enabled or disabled in accordance. If you want to activate all the menus, enter `scgenable(1:2,'on')`.

Modification of the fundamental domain

Three routines let you alter the fundamental domain of an interior map by using a Möbius transformation to modify the solution to the parameter problem.

The functions `hp2disk` and `disk2hp` allow you to interchange solutions for the half-plane and disk interior mapping problems. The function `dfixwc` lets you specify the conformal center (image of zero) of a disk map. A graphical implementation of this function is the routine `ptsource` or the GUI item **Point source**.

Application: Faber polynomials

```
F = faber(m,w,beta,z,c)
```

This M-file uses the Schwarz-Christoffel exterior map to compute the coefficients of Faber polynomials for a polygon. Argument `m` is the degree of the highest-degree polynomial sought. The other input arguments refer to the solution of the exterior mapping problem.

Remarks

Crowding

By far the most troublesome aspect of numerical Schwarz-Christoffel mapping is the *crowding phenomenon*. Elongated polygons have prevertices which are spaced exponentially close, and the parameter problem becomes difficult to solve numerically. If one of the parameter problem solvers repeatedly issues the message `Warning: Severe crowding`, the solution will probably not be found. If you experience this problem, try using the initial-guess argument for *xx*`param`, or subdivide the polygon. If you are working with an interior map, you can also switch between the disk and the half-plane.

A more general approach to crowding is to choose a more suitable fundamental domain. For polygons that are elongated in only one direction, a method based on an infinite strip has been derived by Howell and Trefethen [7], and this method will be incorporated into a version of the SC Toolbox to be released by the end of the summer of 1994.

Conformal center

Unlike SCPACK, the SC Toolbox does not let you directly specify the conformal center (image of zero) when solving the parameter problem for the disk, since three prevertices are fixed rather than just one. Instead, you can solve the parameter problem using `dparam` and then use `dfixwc` to find a map with any given conformal center.

Vertex ordering

The vertices in `w` must be ordered with a positive orientation with respect to the target region. This means a counterclockwise sense for interiors and a clockwise one for exteriors. Use `flipud` to reverse the ordering and `scangle` to update the turning angles.

As a convenience, the GUI will automatically reverse the vertex ordering when you select a parameter problem solution, if all the vertices are finite and seem to have the wrong orientation.

Further reading

For more on the basic Schwarz-Christoffel transformation, see [1, 4, 8]. Several variations, extensions, and applications can be found in [2, 3, 6, 7, 11].

The SC Toolbox implements and extends the methods described by Trefethen in his appendix to [8]. For more on the design of SCPACK, which is numerically similar, see [5, 9, 10].

Suggestions and bug reports

Remember: Anything free comes with no guarantee! I have tried to make the SC Toolbox robust, but I do not explicitly or implicitly warrant its accuracy or reliability. Also, The MathWorks, Inc. is not in any way responsible for the SC Toolbox's design or maintenance.

I welcome complaints, suggestions, and bug reports related to any aspect of the SC Toolbox. I can be reached on the Internet as `driscoll@cam.cornell.edu`. Feel free to report behavior differing from the documentation, make suggestions concerning the graphical user interface, or describe an interesting application you have found for the toolbox.

You are free to change or add to the toolbox's code. However, I request that when distributing the toolbox to others, you package an unmodified version separately from your alterations. If you have code that you think would benefit others, please send it to me and I will consider including it in future versions of the toolbox.

Acknowledgments

This material is based upon work supported by a National Science Foundation Graduate Research Fellowship, by NSF Grant DMS-9116110, and by DOE grant DE-FG02-9YER25199. (Any opinions, findings, conclusions or recommendations expressed in this publication by the author do not necessarily reflect the views of the National Science Foundation.)

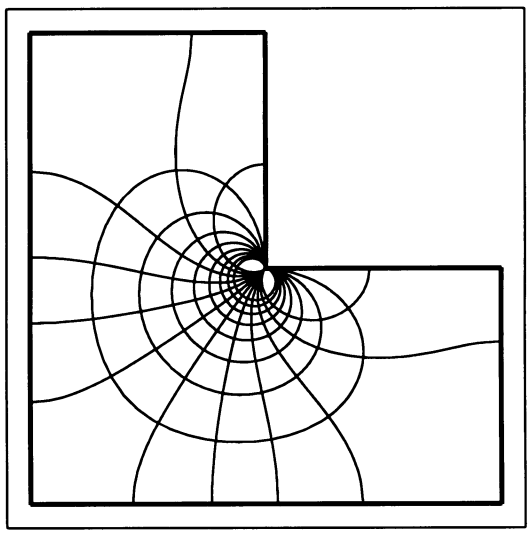
I would like to thank Nick Trefethen for leading me to this project, guiding and encouraging me throughout its development, and serving as chief program tester.

Examples

Simple interior maps

First, explore the half-plane map onto the interior of an L-shaped region.

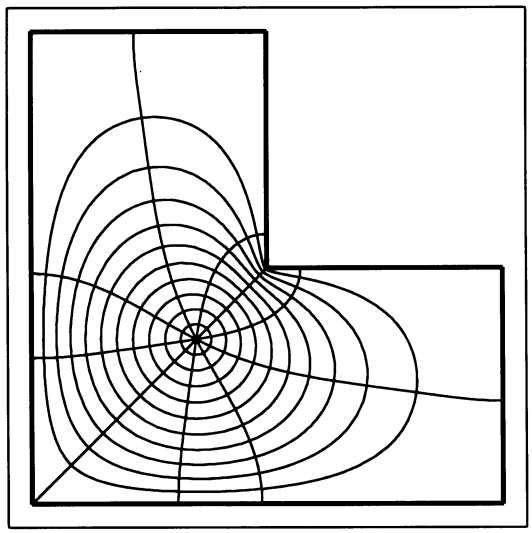
```
>> w=[i;-1+i;-1-i;1-i;1;0];  % define vertices
>> beta=scangle(w);          % compute turning angles
>> [x,c]=hpparam(w,beta);    % solve parameter problem
>> hpdisp(w,beta,x,c)

          w                 beta              x
 ---------------------------------------------------------
        0 +       1i     -0.50000     -1.000000000000e+00
       -1 -       1i     -0.50000      0.000000000000e+00
       -1 -       1i     -0.50000      6.464101617786e+00
        1 +       1i     -0.50000      1.292820323512e+01
        1 +       0i     -0.50000      1.392820323520e+01
        0 +       0i      0.50000                     Inf

  c = -11.665822 + 2.857208e-15i

>> hpplot(w,beta,x,c,12,8)
```



Compare this to the cover of [8].

Now, map from the unit disk instead.

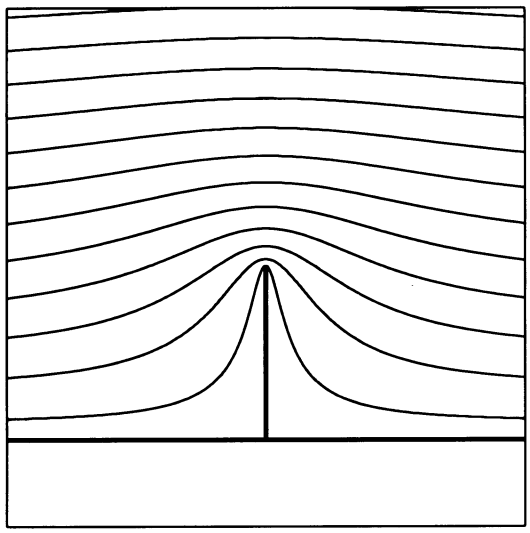
```
>> [z,c]=hp2disk(w,beta,x,c)       % convert to disk

z =

        0 + 1.0000i
  -1.0000
   0.9533 - 0.3022i
   0.9881 - 0.1538i
   0.9897 - 0.1429i
   1.0000

c =

   0.2950 - 0.4878i

>> zp=dinvmap(-.3-.3i,w,beta,z,c) % inverse image of -.3-.3i

zp =

   0.8458 - 0.1672i

>> abs(-.3-.3i - dmap(zp,w,beta,z,c)) % check accuracy

ans =

   5.2972e-15

>> ptsource(w,beta,z,c,-.3-.3i)   % set new conformal center and plot
```



Slits and infinite vertices

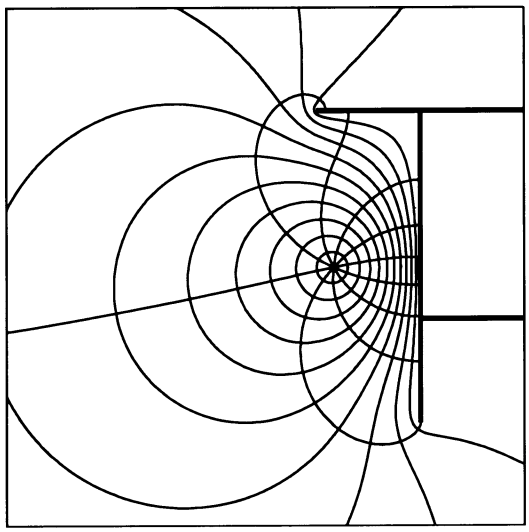
Here are a couple of examples that demonstrate the representations and effects of slits and vertices and infinity.

```
>> w=[0;i;0;Inf];
>> beta=[-.5;1;-.5;-2];
>> [x,c]=hpparam(w,beta);
>> axis([-1.5 1.5 -.5 2.5])  % set axes limits before plotting
>> hold on                   % (and keep them)
>> hpplot(w,beta,x,c,[],0.1:0.2:2.5) % plot streamlines only
```



```
>> w=[-i;-2i;i;-1+i;Inf];    % new region
>> beta=scangle(w)           % compute angles where possible

beta =

       NaN
    1.0000
   -0.5000
       NaN
       NaN

>> beta([1,4,5])=[-.5;1;-3]  % fill in at Inf and its neighbors
```

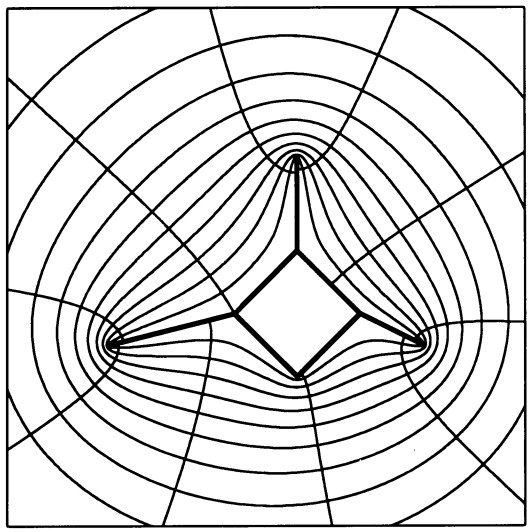
```
beta =

   -0.5000
    1.0000
   -0.5000
    1.0000
   -3.0000

>> [z,c]=dparam(w,beta);
>> axis([-4,1,-3,2])
>> cla
>> dplot(w,beta,z,c,10,10,6)  % higher accuracy to avoid "jaggies"
```



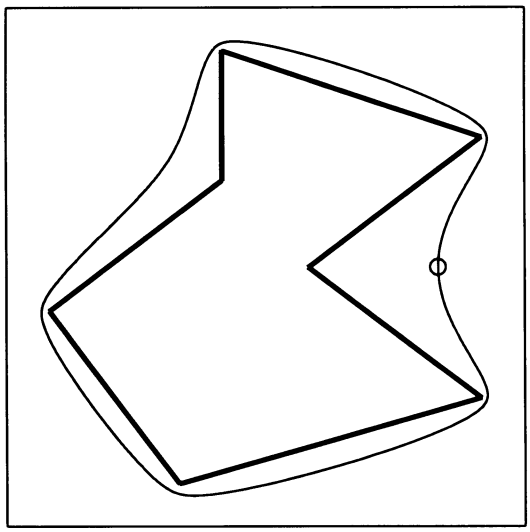
Exterior maps

Exterior mapping works in a similar fashion. Note that slits are now more like spokes.

```
>> [w,beta]=drawpoly;
>> [z,c]=deparam(w,beta);
>> deplot(w,beta,z,c)
```



```
>> [w,beta]=drawpoly;
>> [z,c]=deparam(w,beta);
>> zp=deinvmap(2,w,beta,z,c);
>> deplot(w,beta,z,c,abs(zp),[])  % plot level curve through 2
>> axis([-3,3,-3,3])
>> hold on
>> plot(2+eps*i,'o')              % check it
```



References

[1] R. V. CHURCHILL AND J. W. BROWN, *Complex Variables and Applications*, McGraw-Hill, 5th ed., 1990.

[2] R. T. DAVIS, *Numerical methods for coordinate generation based on Schwarz-Christoffel transformations*, in 4th AIAA Comput. Fluid Dynamics Conf., Williamsburg, VA, 1979, pp. 1–15.

[3] J. M. FLORYAN AND C. ZEMACH, *Schwarz-Christoffel mappings: A general approach*, J. Comput. Phys. 72 (1987), pp. 347–371.

[4] P. HENRICI, *Applied and Computational Complex Analysis*, vol. 1, Wiley, 1974.

[5] ——, *Applied and Computational Complex Analysis*, vol. 3, Wiley, 1986.

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

[7] L. H. HOWELL AND L. N. TREFETHEN, *A modified Schwarz-Christoffel transformation for elongated regions*, SIAM J. Sci. Stat. Comput. 11 (1990), pp. 928–949.

[8] E. B. SAFF AND A. D. SNIDER, *Fundamentals of Complex Analysis*, Prentice Hall, 2nd ed., 1993.

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

[10] ——, *SCPACK user's guide.* MIT Numerical Analysis Report 89-2, 1989.

[11] ——, *Schwarz-Christoffel mapping in the 1980's.* Cornell University Computer Science Department Technical Report TR 93-1381, 1993.