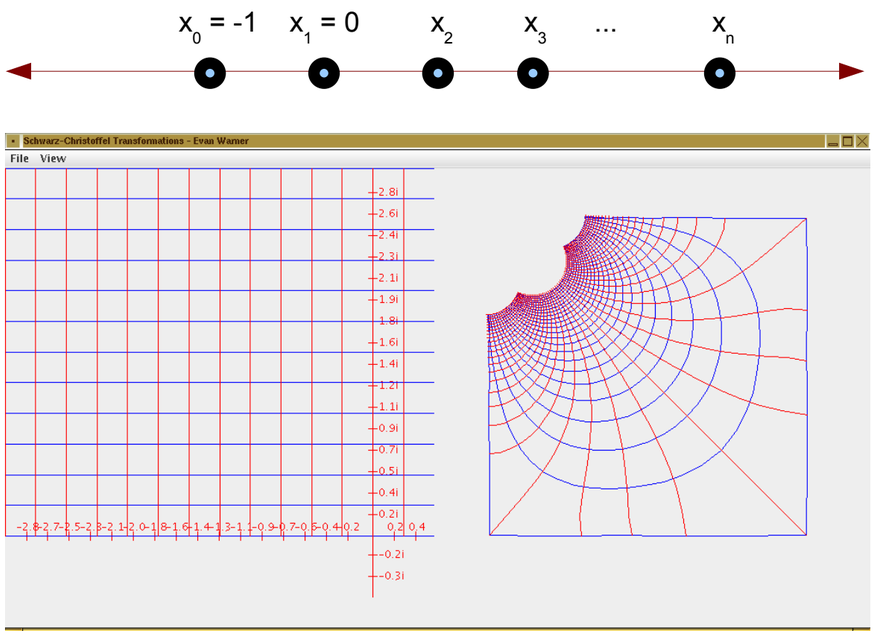
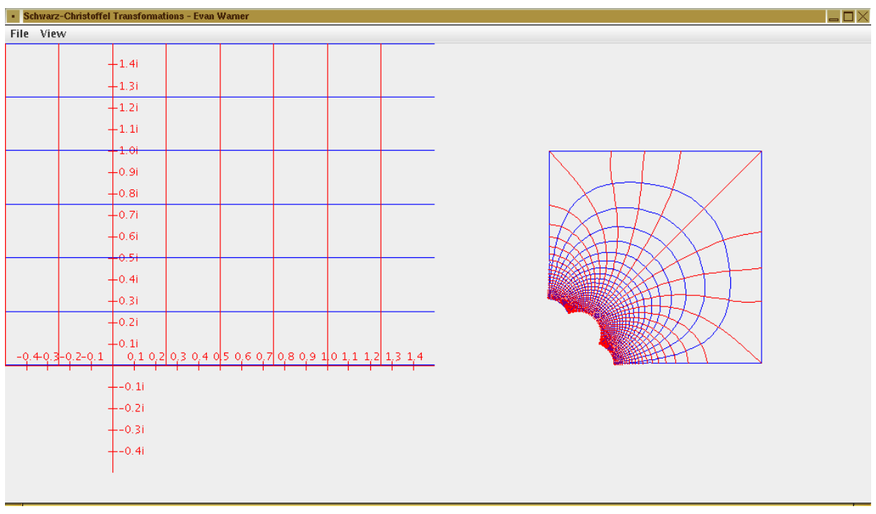
Conformal Mapping Using the Schwarz-Christoffel Transform

Evan Warner – Computer Systems Laboratory 2007–2008





Summary of process

Calculation of prevertices using Newton–Raphson method

↓

Computation of geometrically similar shape

↓

Calculation of auxiliary constants

Development

The software is written entirely in Java, with testing in MATLAB. The entire development of the program is designed to be achieved in stages by attacking the subproblems individually. Several important classes are described below:

- class Complex - this class stores and performs arithmetic on complex numbers, which are not directly supported by Java. Several of the methods, including the multiplication and division algorithms, are designed to run as quickly as possible while avoiding intermediate overflow and floating-point error propagation. The multiplication method, for instance, requires only three real multiplications rather than four.
- class GaussJacobiWeights - this class calculates and stores the sample points and weights for a given Gauss-Jacobi quadrature over the interval $[-1, 1]$. This routine uses Newton's Method to find the roots of the Jacobi polynomials, which are the sample points for the integral, and was taken and translated from [4].
- class SchwarzFunction - this class evaluates the integrand of a given real-valued Schwarz-Christoffel integral, serving as a storage class for data of this kind.
- class GaussQuad - this class accepts as input ψ, a, b, α, and β from Eq. (4). For an arbitrary integral in that form, shifting and scaling the bounds produces the equivalent integral

$$c^{\alpha+\beta+1} \int_{-1}^{1} (\zeta - 1)^{\alpha} (\zeta + 1)^{\beta} \psi(c\zeta + m) d\zeta, \tag{9}$$

  where $b = \frac{a+b}{2}$ and $c = b - m = m - a$. This integral is then evaluated using the sample points and weights given by the GaussJacobiWeights class and returned. For any GaussQuad object, varying numbers of sample points (and thus varying accuracy) are accepted by its integrate() method.
- class RealNewtonRaphson - this class accepts an array of vertices and calculates the necessary prevertices as well as the constants A and B from Eq. (1). The method employs a standard Newton-Raphson method to solve the Eq. (5). At each step, an approximate Jacobian matrix for the function is calculated using a forward-difference method in each dimension; the step vector is then solved for using an LU factorization on the equation

$$\mathbf{J}\delta\vec{x} = \vec{f}, \tag{10}$$

  where $\mathbf{J}$ represents the Jacobian, $\delta\vec{x}$ the step vector, and $\vec{f}$ the current function vector. Note that by employing a forward-difference method to find the Jacobian, the number of function evaluations can be cut in half, as the current function vector can be reused in the Jacobian calculation.
- class ForwardGaussQuad - this class, using already-calculated values for the prevertices, evaluates the Schwarz-Christoffel integral at a given point. To minimize error caused by the presence of singularities near the path of the integral (the singularities at the endpoints are handled by the Gauss-Jacobi quadrature), the path of integration is divided recursively such that no segment is closer to a singularity than one-half its length, a technique employed in [3]. Such recursive subdivision is known as compound Gauss-Jacobi quadrature.
- class SchwarzChristoffel - this class runs the graphical user interface and calls RealNewton-Raphson and ForwardGaussQuad when necessary. The graph itself has the ability to show axes and manually adjust window parameters.

In future iterations of the project, a new set of routines will be implemented to calculate continuous Schwarz-Christoffel problems. Immediately following from Eq. (3) above, we have

Introduction

Many physical problems are expressed as differential or boundary value problems over a surface. Often, these surfaces are or can be approximated by two-dimensional polygons. In this specific case, one method of determining accurate solutions is by taking the polygonal domain to exist in the complex plane and determining a conformal map, which preserves the structure of Laplace's equation, that restates the problem in a simpler domain, most often the upper half-plane. The new problem, now easy to solve analytically or in closed form, is then mapped back to the original domain. For such polygonal domains, a method of determining the specific transform needed is provided by the following formula, known as the Schwarz-Christoffel transform:

$$f(z) = A \int_0^z \prod_{j=1}^{n} (\zeta - x_j)^{-\theta_j/\pi} d\zeta + B. \tag{1}$$

In this formula, ζ is an independent complex variable in the upper half-plane, the θ_j are the exterior angles of the polygon, the x_j are 'preverticies' of the mapping (given along the real axis), n is the number of verticies of the polygon, and A and B are complex constants that specify the location of the image polygon in the complex plane. The θ_j must satisfy

$$\sum_{j=1}^{n} \theta_j = 2\pi, \tag{2}$$

which ensures the completeness of the image polygon [2]. Unfortunately, the Schwarz-Christoffel formula is not easy to evaluate, and requires both effective integration algorithms and efficient, convergent nonlinear equation solvers. Implementation of such numerical routines is not a trivial problem, and is the goal of this project.

Example mappings produced by the program

Results

Results on polygons eleven and fewer vertices suggest that the program has both a reasonable error tolerance and reasonable computation time requirements. Efficiency has been gained by implementing a two-step integration in ForwardGaussQuad which deals with the real and complex parts of the path integral separately and minimizes the average depth of recursion.

Runtime data for a typical computer