PlgCirMap: A MATLAB toolbox for computing the conformal maps from polygonal multiply connected domains onto circular domains

Mohamed M S Nasser

Department of Mathematics, Statistics and Physics,
Qatar University.
mms.nasser@qu.edu.qa

The complex analysis toolbox: new techniques and perspectives,
Isaac Newton Institute for Mathematical Sciences,
September 9 – September 13, 2019, Cambridge, UK.

13 September 2019

Outline

Outline

Conformal mappings

- Conformal mappings are used to transform two-dimensional domains with complicated boundaries (physical domains) onto domains with simpler boundaries (canonical domains).
- This makes the conformal mappings a powerful tool to solve several problems in the fields of science and engineering since the Laplace equation is invariant under conformal mappings.

Simply connected domains

Riemann Mapping Theorem

Let G be a simply connected domain which is not the whole complex plane and let $\alpha \in G$. Then there exists a unique conformal mapping

$$f \ : \ G \ \rightarrow \ D = \{w \ : \ |w| < 1\}$$

satisfying the conditions

$$f(\alpha) = 0, \quad f'(\alpha) > 0. \qquad (*)$$

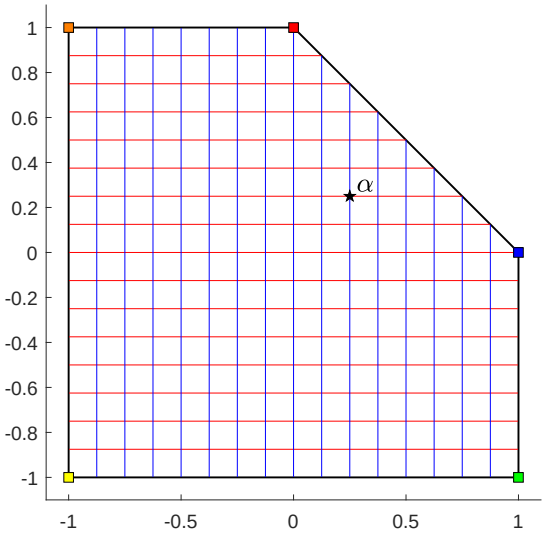
The unit disk $D = \{w \ : \ |w| < 1\}$ is called a canonical domain. The conformal mapping f is also unique if we replace the condition (*) with

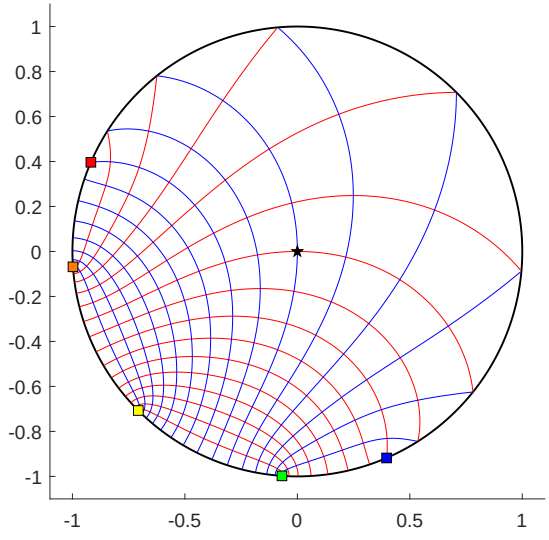
$$f(\alpha) = 0, \quad f(z_1) = 1 \qquad (**)$$

where z_1 is a point on ∂G.

Bounded simply connected domains

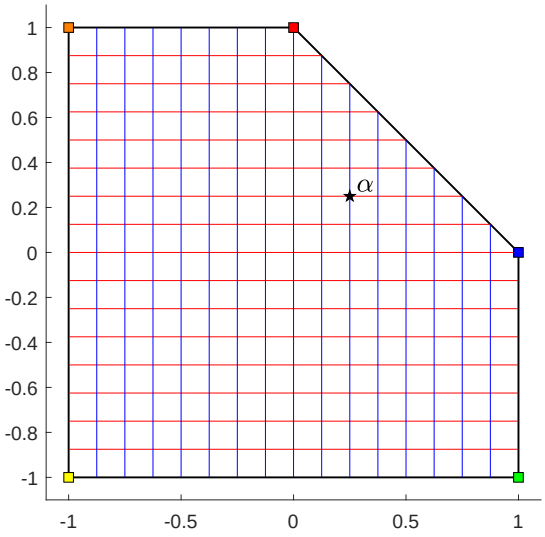
Condition (*)

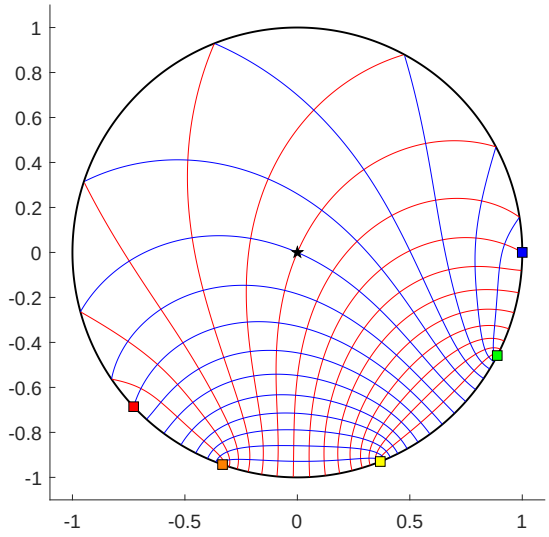




Bounded simply connected domains

Condition (**)





Simply connected domains

The exterior unit disk $D^- = \{w : |w| > 1\}$ is also a canonical domain for the conformal mapping of simply connected domains.

Theorem

Let G be a simply connected domain which is not the whole complex plane and let $\infty \in G$. Then there exists a unique conformal mapping

$$f : G \to D^- = \{w : |w| > 1\}$$

satisfying the conditions

$$f(\infty) = \infty, \quad f'(\infty) > 0. \qquad (*)$$

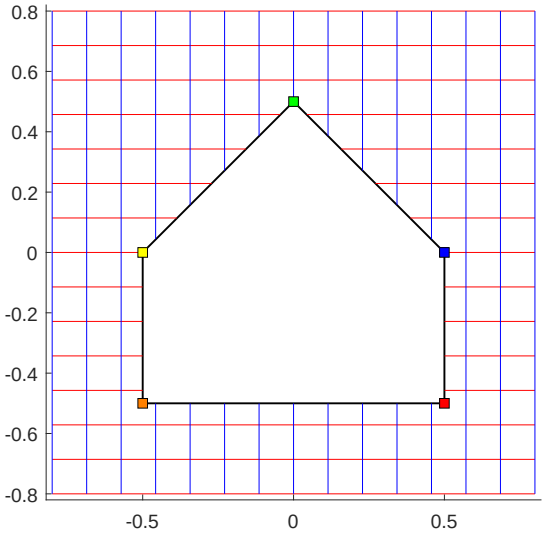
The conformal mapping f is also unique if we replace the condition (*) with

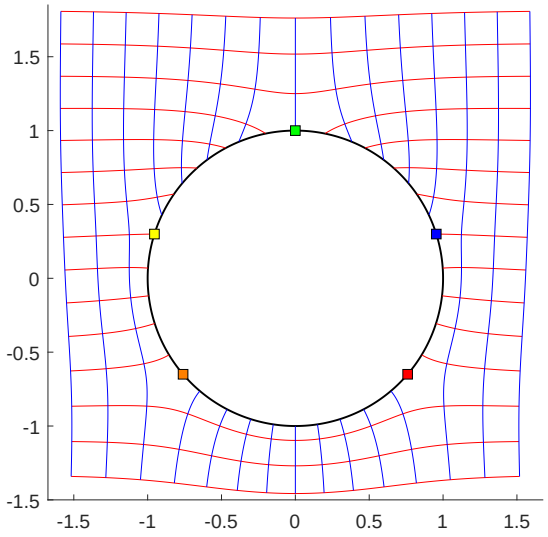
$$f(\infty) = \infty, \quad f(z_1) = 1 \qquad (**)$$

where z_1 is a point on ∂G.

Unbounded simply connected domains

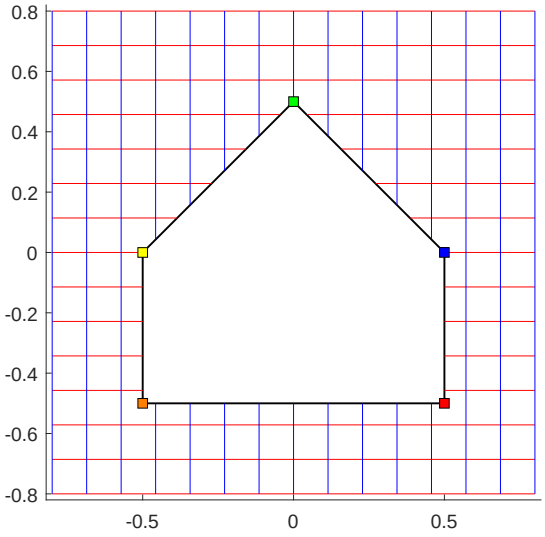
Condition (*)

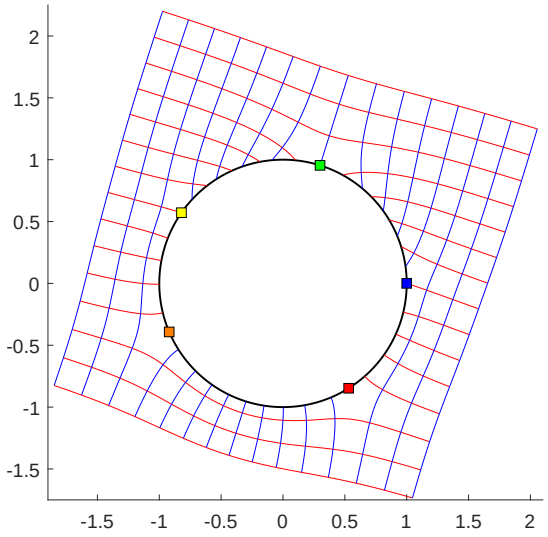




Unbounded simply connected domains

Condition (**)





Multiply connected domains

- Numerous canonical domains have been considered in the literature for conformal mapping of multiply connected domains in the extended complex plane $\mathbb{C} \cup \{\infty\}$.
- Thirty nine canonical slit domains have been catalogued by Koebe in his classical paper

  P. Koebe.
  Abhandlungen zur Theorie der konformen Abbildung, IV. Abbildung mehrfach zusammenhängender schlichter Bereiche auf Schlitzbe-reiche,
  Acta Math. 41 (1918) 305–344.
- The infinite strip with parallel slits.
- The lemniscatic domain.
- Perhaps the most important canonical domain for multiply connected domains is the circular domain.

Why circular domains?

- Circular domains are important from physical and computational points of view.
- Circular domains are ideal domains for using Fourier series and FFT.
- Recently, Professor Darren Crowdy has derived analytic formulas for several problems in multiply connected circular domains. These analytic formulas are described in terms of a special function known as the Schottky-Klein prime function.

Multiply connected domains

Theorem

Let G be a bounded multiply connected domain with $m-1$ holes bordered by $\Gamma = \partial G = \Gamma_1 \cup \cdots \cup \Gamma_m$ where Γ_m is the outer boundary. Then there exists a bounded circular domain D and a conformal mapping $f : G \to D$. Assume D is bordered by $C = \partial D = C_1 \cup \cdots \cup C_m$ where C_m is the outer boundary. Then, both D and f are uniquely determined by the region G when the following normalization conditions are imposed:

- *The outer boundary C_m of D is the unit circle.*
- *For a given point $\alpha \in G$, the function f satisfies*

$$f(\alpha) = 0, \quad f'(\alpha) > 0. \qquad (*)$$

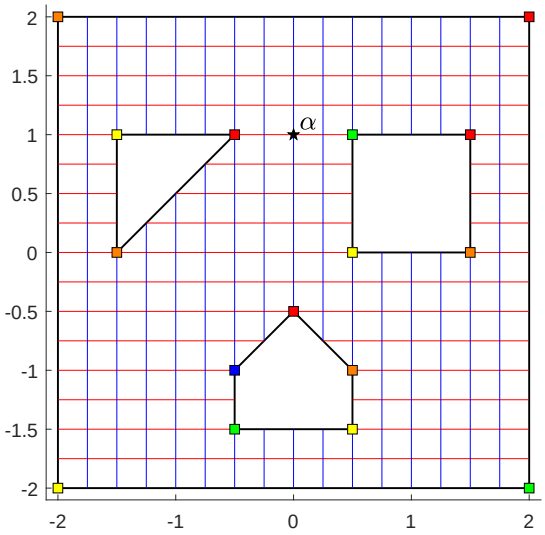
Condition (*) can be replaced with

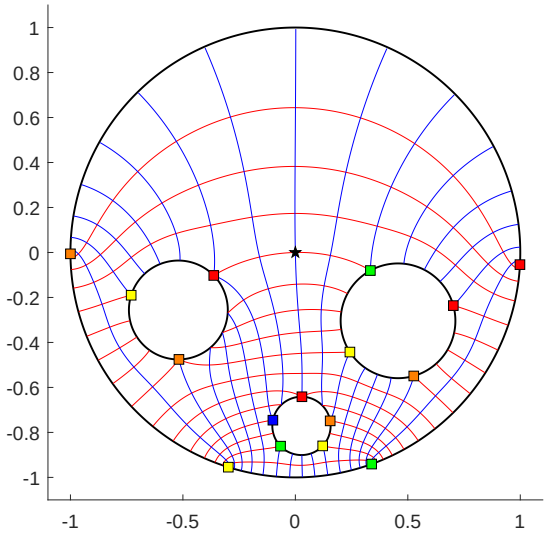
$$f(\alpha) = 0, \quad f(z_1) = 1 \qquad (**)$$

where z_1 is a point on Γ_m.

Bounded multiply connected domains

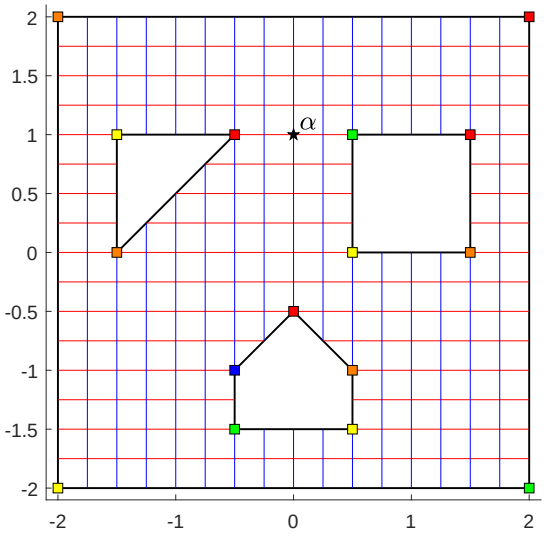
Condition (*)

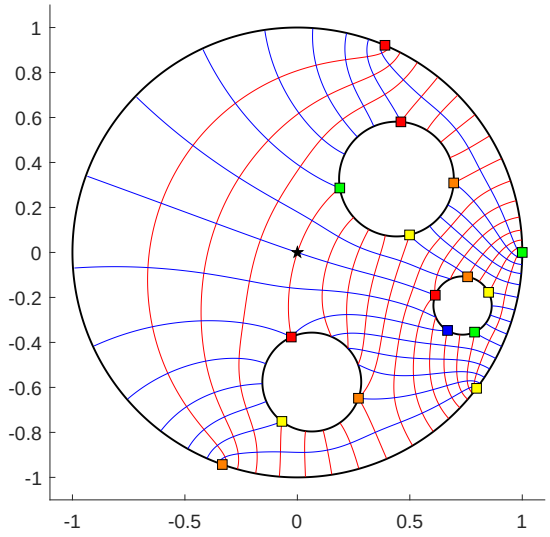




Bounded multiply connected domains

Condition (**)





Multiply connected domains

Theorem

Let G be an unbounded multiply connected domain with m holes. Then there exists a bounded circular domain D and a conformal mapping $f : G \to D$. *Both D and f are uniquely determined by the region G when the following normalization condition is imposed:*

$$f(z) = z + O\left(\frac{1}{z}\right) \quad \text{near } \infty. \tag{*}$$

Assume G bordered by $\Gamma = \partial G = \Gamma_1 \cup \cdots \cup \Gamma_m$ and D is bordered by $C = \partial D = C_1 \cup \cdots \cup C_m$. Condition (*) can be replaced with

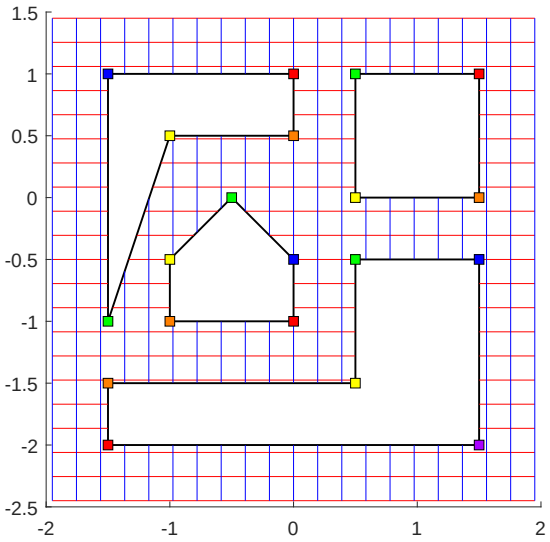
- The boundary C_m is the unit circle.
- The function f satisfies

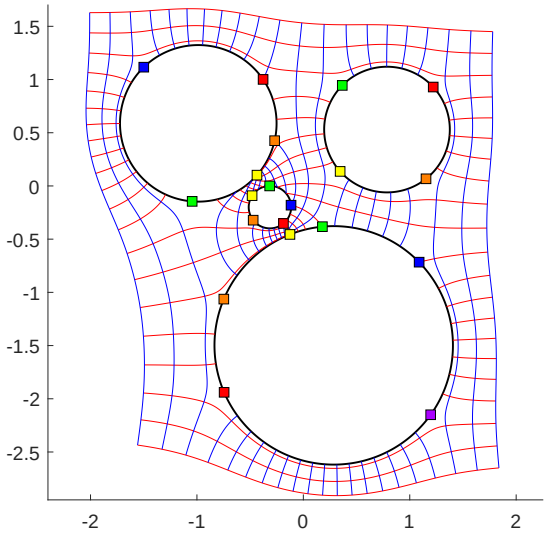
$$f(\infty) = \infty, \quad f(z_1) = 1 \tag{**}$$

where z_1 is a point on Γ_m.

Unbounded multiply connected domains

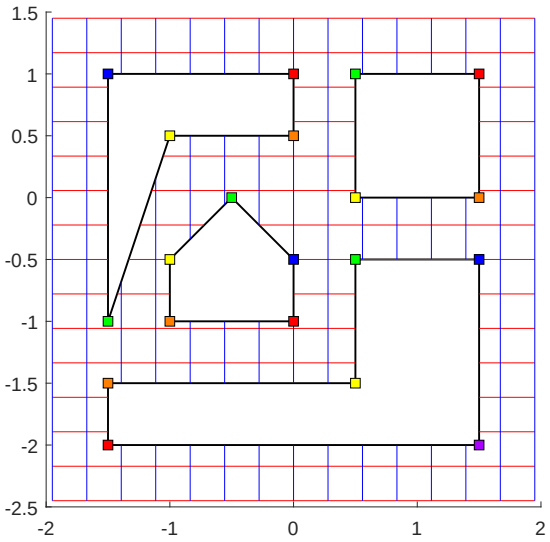
Condition (*)

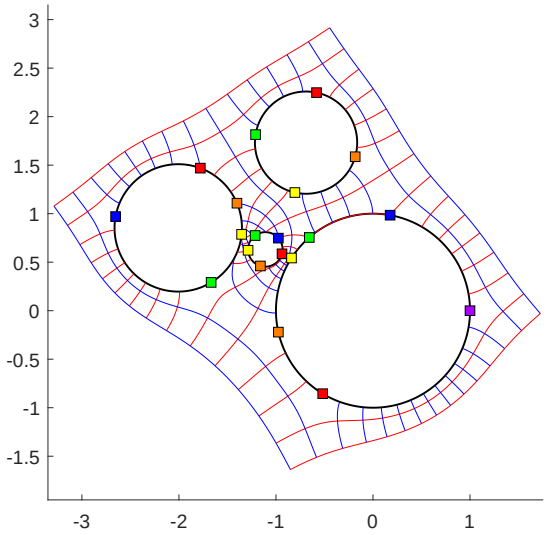




Unbounded multiply connected domains

Condition (**)





Schwarz-Christoffel mapping

- The stranded method for computing the conformal mapping from circular domains to domains bordered by polygons is to use the Schwarz-Christoffel mapping.
- For simply connected domains, the Schwartz-Christoffel formula provide us with an explicit formula for computing the conformal mapping from the unit disk to simply connected polygonal domains.

T. A. Driscoll and L. N. Trefethen.
Schwarz-christoffel mapping.
Cambridge University Press, Cambridge, 2002.

The Schwarz-Christoffel toolbox

The Schwarz-Christoffel toolbox:

T. A. Driscoll.
Schwarz-Christoffel Toolbox, Version 2.4.1.
github.com/tobydriscoll/sc-toolbox.

has been widely used by many researchers in applications require the numerical computing of the Schwarz-Christoffel mapping for simply connected domains.

Multiply connected domains

- D.G. Crowdy.
  The Schwarz–Christoffel mapping to bounded multiply connected domains.
  Proc. Royal Soc. A 461 (2005) 2653–2678.
- D.G. Crowdy.
  Schwarz–Christoffel mappings to unbounded multiply connected polygonal regions.
  Math. Camb. Phil. Soc. 142 (2007) 319–339.
- T. K. DeLillo, A. R. Elcrat and J. A. Pfaltzgraff.
  Schwarz-Christoffel mapping of multiply connected domains.
  J. d'Anal. Math. 94 (2004) 17–47.
- T.K. DeLillo.
  Schwarz-Christoffel mapping of bounded multiply connected domains.
  Comput. Methods Funct. Theory 6 (2006) 275–300.

Multiply connected domains

- T.K. DeLillo, T.A. Driscoll, A.R. Elcrat, and J.A. Pfaltzgraff.
  Computation of multiply connected Schwarz-Christoffel maps for exterior domains.
  Comput. Methods Funct. Theory 6 (2006) 301–315.
- T.K. Delillo and E.H. Kropf.
  Numerical computation of the Schwarz-Christoffel transformation for multiply connected domains.
  SIAM J. Sci. Comput. 33 (2011) 1369–1394.
- T.K. DeLillo, A.R. Elcrat, E.H. Kropf and J.A. Pfaltzgraff.
  Efficient calculation of Schwarz-Christoffel transformations for multiply connected domains using Laurent series.
  Comput. Methods Funct. Theory 13 (2013) 307–336.

PlgCirMap toolbox

The method used to write the PlgCirMap toolbox is based on the fast implementation of Koebe's iterative method which presented in:

M.M.S. Nasser.
Fast Computation of the Circular Map.
Comput. Methods Funct. Theory 15 (2015) 187–223.

Outline

Parametrization the boundary

- Let G be a multiply connected domain bordered by the polygons

$$\Gamma = \partial G = \Gamma_1 \cup \cdots \cup \Gamma_m$$

  where Γ_m is the outer boundary if G is bounded. We assume the interior angles at the vertices of the polygons are in $(0, 2\pi)$.
- For $k = 1, 2, \ldots, m$, we assume the vertices of Γ_k are:

$$v_{k,1}, \quad v_{k,2}, \quad v_{k,3}, \quad \ldots, \quad v_{k,\ell_k}.$$

- These vertices are assumed to be ordered in clockwise orientation except for the case Γ_m is the outer boundary. In the later case, the vertices of Γ_m are ordered in in counterclockwise orientation.

Parametrization the boundary component Γ_k

- We parametrize the polygon Γ_k by the 2π-periodic function

$$\hat{\eta}_k(t) = \begin{cases} v_{k,1} + \frac{\ell_k}{2\pi}\left(v_{k,2} - v_{k,1}\right) t, & t \in [0, 2\pi/\ell_k), \\ v_{k,2} + \frac{\ell_k}{2\pi}\left(v_{k,3} - v_{k,2}\right)(t - 2\pi/\ell_k), & t \in [2\pi/\ell_k, 4\pi/\ell_k), \\ \quad \vdots & \\ v_{k,\ell_k-1} + \frac{\ell_k}{2\pi}\left(v_{k,\ell_k} - v_{k,\ell_k-1}\right)(t - 2(\ell_k-2)\pi/\ell_k), & \\ & t \in [2(\ell_k-2)\pi/\ell_k, 2(\ell_k-1)\pi/\ell_k), \\ v_{k,\ell_k} + \frac{\ell_k}{2\pi}\left(v_{k,1} - v_{k,\ell_k}\right)(t - 2(\ell_k-1)\pi/\ell_k), & \\ & t \in [2(\ell_k-1)\pi/\ell_k, 2\pi]. \end{cases}$$

- Note that $\hat{\eta}_k'(t)$ is not continuous at the corner points.

Parametrization the boundary component Γ_k

- For domains with corners, the integral operator with the generalized Neumann kernel is not compact. But, it can be written as a sum of a compact operator and bounded non-compact operator with norm less than one in suitable function spaces. Hence, we can apply the Fredholm theory to the integral equation with the generalized Neumann kernel although the operator is not compact[1].
- The solution of the integral equation has a singularity in its first derivative in the vicinity of the corner points.
- Using the trapezoidal rule with equidistant nodes to discretize the integral equation yields only poor convergence.

[1] M.M.S. Nasser, A.H.M. Murid and Z. Zamzamir, A boundary integral method for the Riemann–Hilbert problem in domains with corners, Complex Var. Elliptic Equ. 53(11) (2008) 989–1008.

Parametrization the boundary component Γ_k

- To achieve a satisfactory accuracy, we discretize the integral equation using a *graded mesh* which is based on substituting a new variable in such a way that the discontinuity of the derivatives of the solution of the integral equation at the corner points is removed.

Parametrization the boundary component Γ_k

- Consider the bijective, strictly monotonically increasing and infinitely differentiable function, $w : [0, 2\pi] \to [0, 2\pi]$, defined by[2]

$$w(t) = 2\pi \frac{v(t)^p}{v(t)^p + v(2\pi - t)^p},$$

where

$$v(t) = \left(\frac{1}{p} - \frac{1}{2}\right)\left(\frac{\pi - t}{\pi}\right)^3 + \frac{1}{p}\frac{t - \pi}{\pi} + \frac{1}{2},$$

where the integer $p \geq 2$ is called the grading parameter.

- Note that

$$w'(0) = \cdots = w^{(p-1)}(0) = 0$$

and

$$w'(2\pi) = \cdots = w^{(p-1)}(2\pi) = 0.$$

[2] R. Kress, A Nyström method for boundary integral equations in domains with corners, Numer. Math. 58(2) (1990) 145–161.

Parametrization the boundary component Γ_k

- We define a bijective, strictly monotonically increasing function, $\delta_k : [0, 2\pi] \to [0, 2\pi]$, by[3]

$$\delta_k(t) = \begin{cases} \frac{w(\ell_k t)}{\ell_k}, & t \in \left[0, \frac{2\pi}{\ell_k}\right), \\ \frac{w(\ell_k t - 2\pi) + 2\pi}{\ell_k}, & t \in \left[\frac{2\pi}{\ell_k}, \frac{4\pi}{\ell_k}\right), \\ \frac{w(\ell_k t - 4\pi) + 4\pi}{\ell_k}, & t \in \left[\frac{4\pi}{\ell_k}, \frac{6\pi}{\ell_k}\right), \\ \quad \vdots & \\ \frac{w(\ell_k t - 2(\ell_k - 1)\pi) + 2(\ell_k - 1)\pi}{\ell_k}, & t \in \left[\frac{2(\ell_k - 1)\pi}{\ell_k}, 2\pi\right]. \end{cases}$$

- The function δ_k is at least p times continuously differentiable, and

$$\delta_k'(2\pi j/\ell_k) = \cdots = \delta_k^{(p-1)}(2\pi j/\ell_k) = 0$$

for $j = 0, 1, \ldots, \ell_k$.

[3] J. Liesen, O. Séte and M.M.S. Nasser, Fast and Accurate Computation of the Logarithmic Capacity of Compact Sets, Comput. Methods Funct. Theory 17 (2017) 689–713.

Parametrization the boundary component Γ_k

- Using the graded mesh generated by the functions $\delta_1(t), \ldots, \delta_m(t)$ is equivalent to using these function to parametrize the boundaries $\Gamma_1, \ldots, \Gamma_m$ and then solving the integral equation using the trapezoidal rule with equidistant nodes.[4]
- For $k = 1, 2, \ldots, m$, we then parametrize the polygon Γ_k by:

$$\eta_k(t) = \hat{\eta}_k(\delta_k(t)), \quad t \in [0, 2\pi].$$

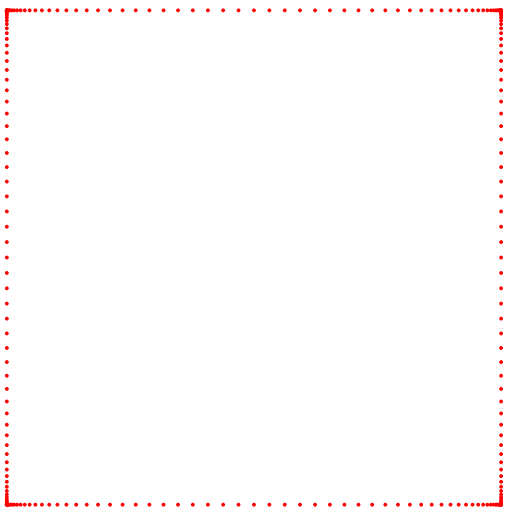
Then

$$\eta_k'(t) = \hat{\eta}_k'(\delta_k(t))\delta_k'(t), \quad t \in [0, 2\pi].$$

[4] R. Kress, Boundary integral equations in time-harmonic acoustic scattering. Math. Comput. Modelling 15(3-5) (1991) 229–243.

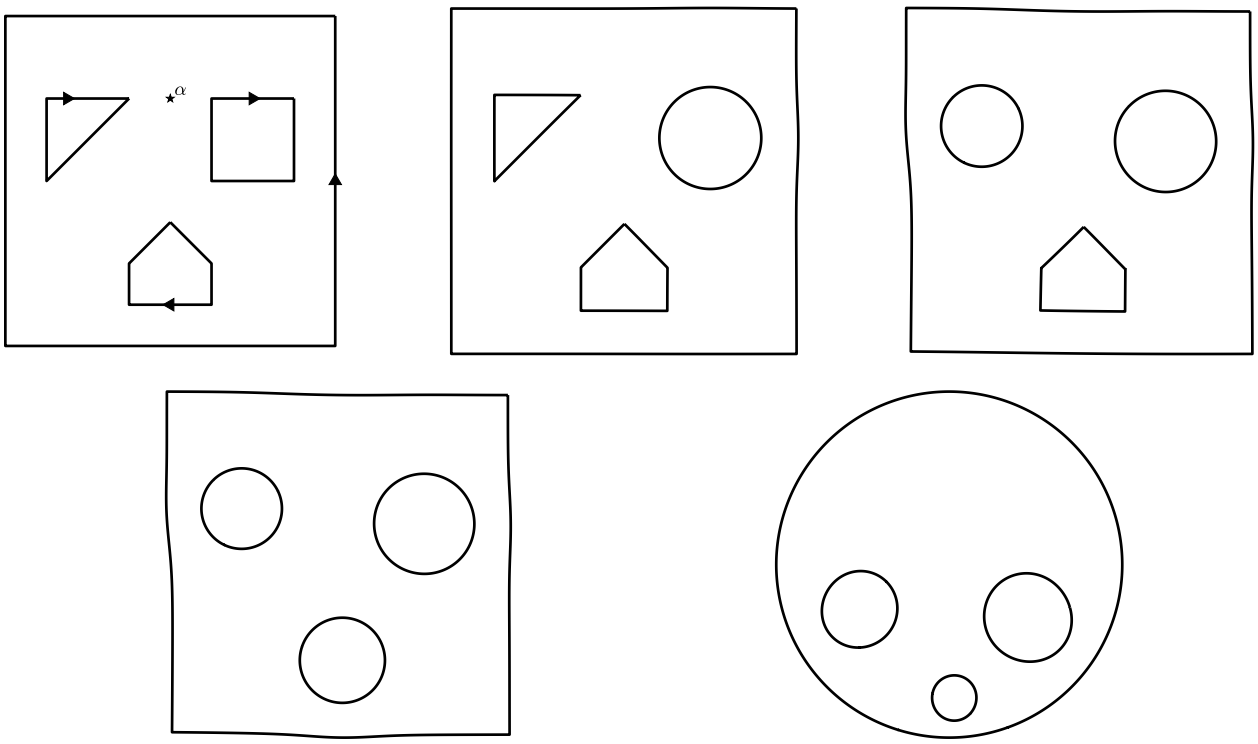
Example

```
ver = [ 2+2i; -2+2i; -2-2i; 2-2i];
[et,etp]=polygonp(ver,64);
plot(et,'.r');
axis equal; axis off
```

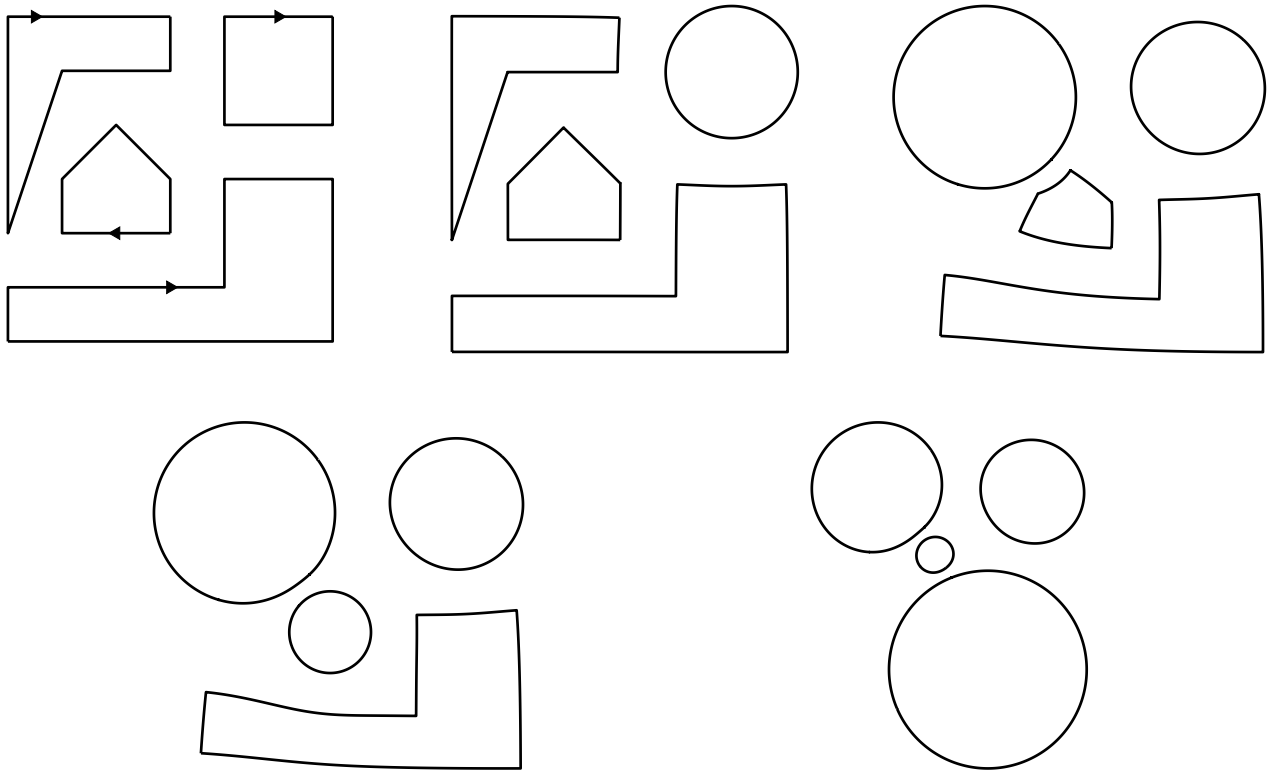


Outline

Koebe's iterative method-bounded domain



Koebe's iterative method-unbounded domain



The function A_k

- For $k = 1, 2, \ldots, m-1$, we define a function $A_k(t)$, $t \in [0, 2\pi]$, by

$$A_k(t) = 1.$$

- For $k = m$, we define a function $A_m(t)$, $t \in [0, 2\pi]$, by

$$A_m(t) = \begin{cases} \eta_m(t) - \alpha, & \text{if } G \text{ is bounded,} \\ 1, & \text{if } G \text{ is unbounded,} \end{cases}$$

where α is a given point in G.

The generalized Neumann kernel

- For $k = 1, 2, \ldots, m$, the generalized Neumann kernel N_k formed with A_k and η_k is defined by[5]

$$N_k(s,t) = \frac{1}{\pi} \operatorname{Im}\left(\frac{A_k(s)}{A_k(t)} \frac{\eta_k'(t)}{\eta_k(t) - \eta_k(s)} \right).$$

- We define also a kernel M_k by

$$M_k(s,t) = \frac{1}{\pi} \operatorname{Re}\left(\frac{A_k(s)}{A_k(t)} \frac{\eta_k'(t)}{\eta_k(t) - \eta_k(s)} \right).$$

- The kernel N_k is continuous and the kernel M_k is singular with

$$M_k(s,t) = -\frac{1}{2\pi} \cot\frac{s-t}{2} + \hat{M}_k(s,t)$$

where $\hat{M}_k$ is continuous.

[5] R. Wegmann, A.H.M. Murid and M.M.S. Nasser, The Riemann-Hilbert problem and the generalized Neumann kernel. J. Comput. Appl. Math. 182 (2005) 388–415.

The generalized Neumann kernel

Then, we define the integral operators

$$\mathbf{N}_k\mu(s) = \int_0^{2\pi} N_k(s,t)\mu(t)dt, \quad s \in [0, 2\pi],$$

and

$$\mathbf{M}_k\mu(s) = \int_0^{2\pi} M_k(s,t)\mu(t)dt, \quad s \in [0, 2\pi],$$

for $k = 1, 2, \ldots, m$.

Conformal mapping from the exterior of Γ_k

To compute the unique conformal mapping $w = \Phi(z)$ from the exterior domain of Γ_k onto the exterior unit disk with the normalization[6]

$$\Phi(\infty) = \infty, \quad \Phi'(\infty) > 0,$$

let

- $\gamma_k(t) = -\log|\eta_k(t) - \alpha_k|$ (α_k is an auxiliary point interior to Γ_k).
- $(\mathbf{I} - \mathbf{N}_k)\mu_k = -\mathbf{M}_k\gamma_k$.
- $c = e^{-h_k}$ where $h_k = [\mathbf{M}_k\mu_k - (\mathbf{I} - \mathbf{N}_k)\gamma_k]/2$.
- $A_k(t)g(\eta_k(t)) = \gamma_k(t) + \mathrm{i}\mu_k(t)$.

Then

$$\Phi(z) = c\,(z - \alpha_k)\,e^{g(z)}.$$

[6] M.M.S. Nasser, Fast Computation of the Circular Map, Comput. Methods Funct. Theory 15 (2015) 187–223.

Conformal mapping from the interior of Γ_m

If G is bounded, to compute the unique conformal mapping $w = \Phi(z)$ from the interior domain of Γ_m onto the unit disk with the normalization

$$\Phi(\alpha) = 0, \quad \Phi'(\alpha) > 0,$$

let

- $\gamma_m(t) = -\log|\eta_m(t) - \alpha|$.
- $(\mathbf{I} - \mathbf{N}_m)\mu_m = -\mathbf{M}_m\gamma_m$.
- $c = e^{-h_m}$ where $h_m = [\mathbf{M}_m\mu_m - (\mathbf{I} - \mathbf{N}_m)\gamma_m]/2$.
- $A_m(t)g(\eta_m(t)) = \gamma_m(t) + \mathrm{i}\mu_m(t)$.

Then

$$\Phi(z) = c\,(z - \alpha)\,e^{(z-\alpha)g(z)}.$$

Solving the integral equation

A MATLAB function `fbie` for fast computing the functions μ_k and h_k in $O(n_k \ln n_k)$ operations is given in:

M.M.S. Nasser.
Fast solution of boundary integral equations with the generalized Neumann kernel,
Electron. Trans. Numer. Anal. 44 (2015) 189–229.

In the MATLAB function `fbie`, we use the function `zfmm2dpart` from the MATLAB toolbox:

L. Greengard and Z. Gimbutas.
FMMLIB2D: A MATLAB toolbox for fast multipole method in two dimensions,
Version 1.2, 2012. `http://www.cims.nyu.edu/cmcl/fmm2dlib/fmm2dlib.html`.

Outline

MATLAB files

A collection of MATLAB files have integrated together to write a friendly MATLAB toolbox (in preliminary stage) for computing the conformal mapping from bounded/unbounded multiply connected polygonal domains onto circular domains.
These files can be downloaded from:
`https://github.com/mmsnasser/plgcirmap`
The three main functions in this toolbox are:

- `plgcirmap`.
- `plotmap`.
- `evalu`.

`plgcirmap`

To compute the conformal mapping with the normalization

$$f(\alpha) = 0, \quad f'(\alpha) > 0$$

for bounded G; and

$$f(z) = z + O\left(\frac{1}{z}\right) \quad \text{near } \infty,$$

for unbounded G, we call the function:

```
f=plgcirmap(ver,alpha);
```

where

- `ver`: A cell array where `ver{k}` is a vector of the vertices of the polygon Γ_k, $k = 1, 2, \ldots, m$.
- `alpha`: A given point in the domain G if G is bounded and `alpha=inf` if G is unbounded.

plgcirmap

To compute the conformal mapping with the normalization

$$f(\alpha) = 0, \quad f(z_1) = 1,$$

for bounded G; and

$$f(\infty) = \infty, \quad f(z_1) = 1,$$

for unbounded G, we call the function:

```
f=plgcirmap(ver,alpha,ver{end}(end));
```

where the vertices of the polygon Γ_m are ordered such that z_1 is the last element in the vector `ver{m}`; i.e., z_1 =`ver{end}(end)` is the last vertex of the last polygon.

plgcirmap

`f` is an object with:

- `f.ver`: the vertices of the polygons.
- `f.imgver`: a cell array where `f.imgver{k}`, `k=1,2,...,m`, contains the image of the vertices `ver{k}` under the conformal mapping.
- `f.nv`: a vector of length `m` where `nv(k)` = `f.nv(k)` is the number of nodes on the boundary component Γ_k, $k = 1, 2, ..., m$.
- `f.et`: the parametrization of the boundary of the polygonal domain G; i.e.,
  `et=f.et` is a vector with `m*(nv(1)+···+nv(m))` points where
  `et(1:nv(1))` is the discretization of Γ_1,
  `et(1+nv(1):nv(1)+nv(2))` is the discretization of Γ_2, ... etc.
- `f.etp`: the first derivative of the parametrization of the boundary of the polygonal domain G.

plgcirmap

- `f.zet`: the parametrization of the boundary of the circular domain G, i.e.,
  `zet=f.zet` is a vector with `m*(nv(1)+···+nv(m))` points where `zet(1:nv(1))` is the discretization of C_1,
  `zet(1+nv(1):nv(1)+nv(2))` is the discretization of C_2, ... etc.
- `f.zetp`: the first derivative of the parametrization of the boundary of the circular domain G.
- `f.cent`: a vector of length `m` where `f.cent(k)` is the center of the circle C_k, $k = 1, 2, ..., m$.
- `f.rad`: a vector of length `m` where `f.rad(k)` is the radius of the circle C_k, $k = 1, 2, ..., m$.
- `f.alpha`: the point `alpha`.
- `f.inf`: (only for unbounded G where `f.inf=f'(inf)`.

plgcirmap

```
iprec        = 4; % the accuracy of the FMM method is 0.5e-12
gmresrestart = []; % the GMRES method is used without restart
gmrestol     = 0.5e-12; % tolerances of the GMRES method
gmresmaxit   = 100; % maximum number of iterations for the ...
    GMRES method
koebemaxit   = 100; % maximum number of iterations for the ...
    Koebe method
koebetol     = 1e-12; % tolerances of the Koebe method
```

plotmap

The function `plotmap` is used to plot the conformal map f as following:

```
plotmap(f,'d','rec',n1,n2);
```

plots rectangular grids (`n1` horizontal lines and `n2` vertical lines) in the domain G and their images in the circular domain D under the conformal map.

```
plotmap(f,'v','rec',n1,n2);
```

plots rectangular grids (`n1` horizontal lines and `n2` vertical lines) in the domain D and their images in the domain G under the inverse conformal map.

plotmap

```
plotmap(f,'d','plr',n1,n2);
```

plots polar grids (`n1` circles and `n2` rays) in the domain G and their images in the circular domain D under the conformal map.

```
plotmap(f,'v','plr',n1,n2);
```

plots polar grids (`n1` circles and `n2` rays) in the domain D and their images in the domain G under the inverse conformal map.

```
plotmap(f);
```

plots in the polygonal domain G and the circular domain D.

The function `evalu` is used to compute the value of the mapping function f or its inverse at a given vector of points. The function can be called as:

```
w = evalu(f,z,b);
```

where

- `b='d'` for computing the values `w` of the mapping function at a row vector of points `z` in the polygonal domain G.
- `b='v'` for computing the values `w` of the invers mapping function at a row vector of points `z` in the circular domain D.

Outline

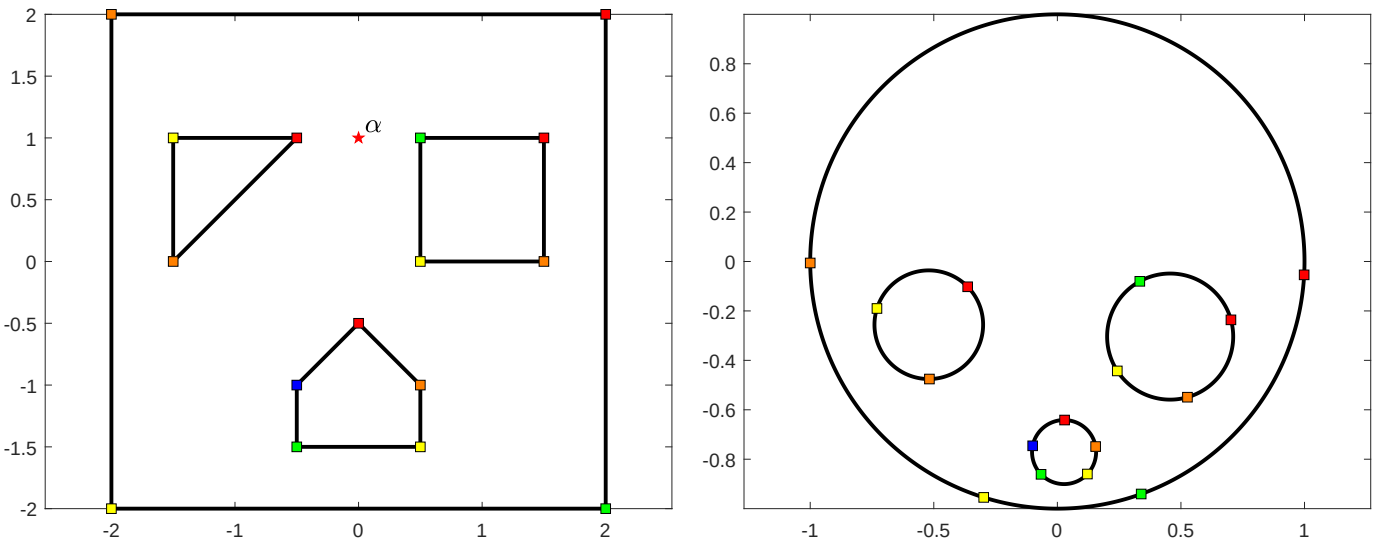
Bounded multiply connected domain $m = 4$

```
clc; clear all;
ver{1}=[ 1.5+1.0i ;  1.5+0.0i ;  0.5+0.0i ; 0.5+1.0i];
ver{2}=[-0.5+1.0i ; -1.5+0.0i ;-1.5+1.0i];
ver{3}=[-0.0-0.5i ;  0.5-1.0i ; 0.5-1.5i  ;-0.5-1.5i ...
    ;-0.5-1.0i];
ver{4}=[ 2.0+2.0i ; -2.0+2.0i ; -2.0-2.0i ; 2.0-2.0i];
alpha = i;
```

```
f=plgcirmap(ver,alpha);
```

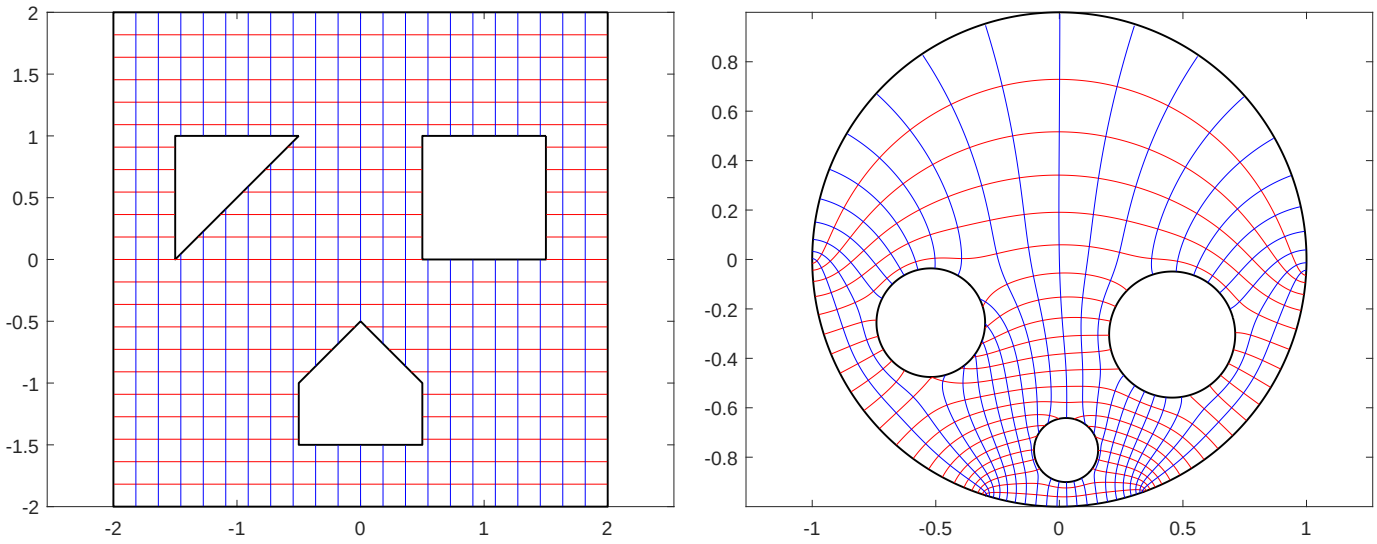

Bounded multiply connected domain $m = 4$

```
plotmap(f);
```



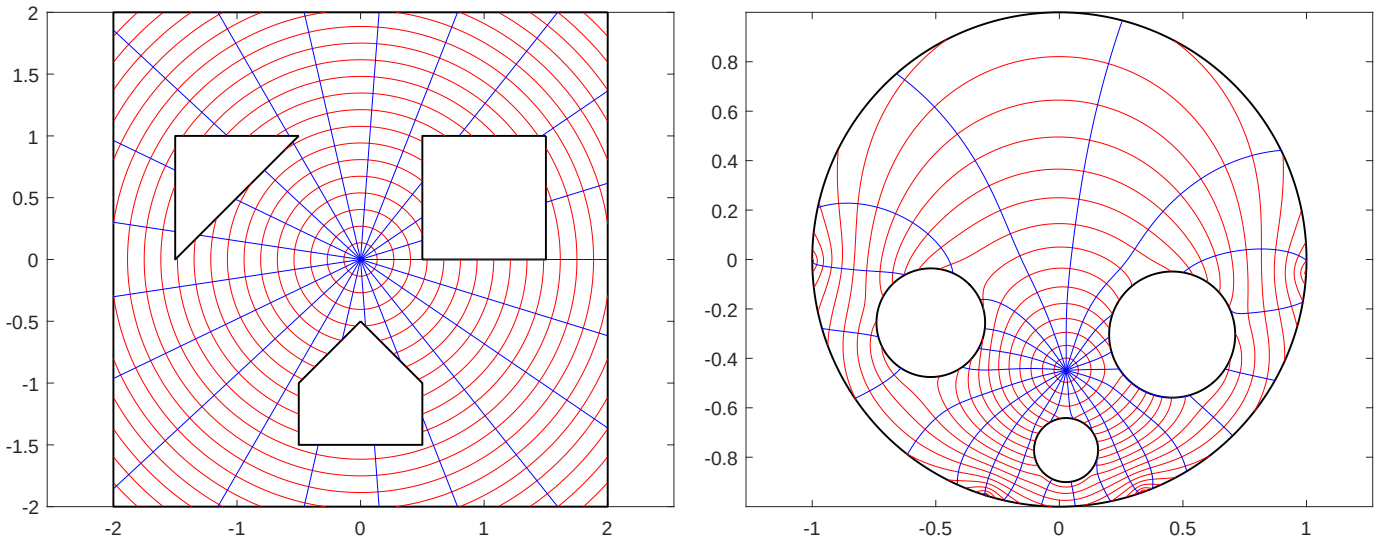
Bounded multiply connected domain $m = 4$

```
plotmap(f,'d','rec',21,21);
```



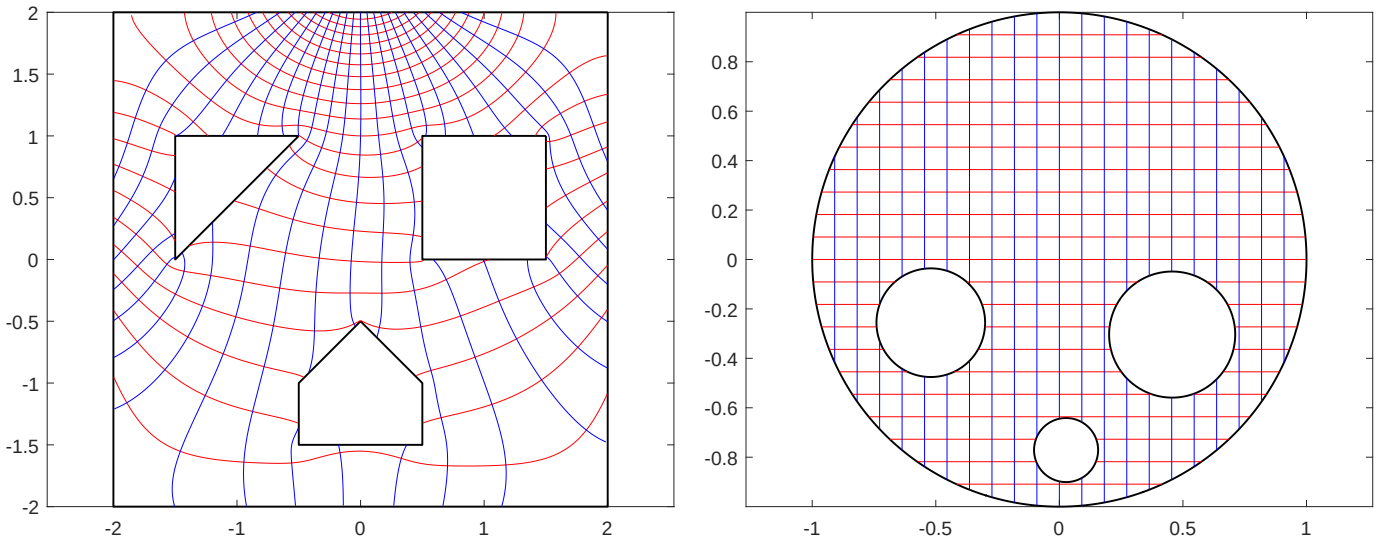
Bounded multiply connected domain $m = 4$

```
plotmap(f,'d','plr',21,21);
```



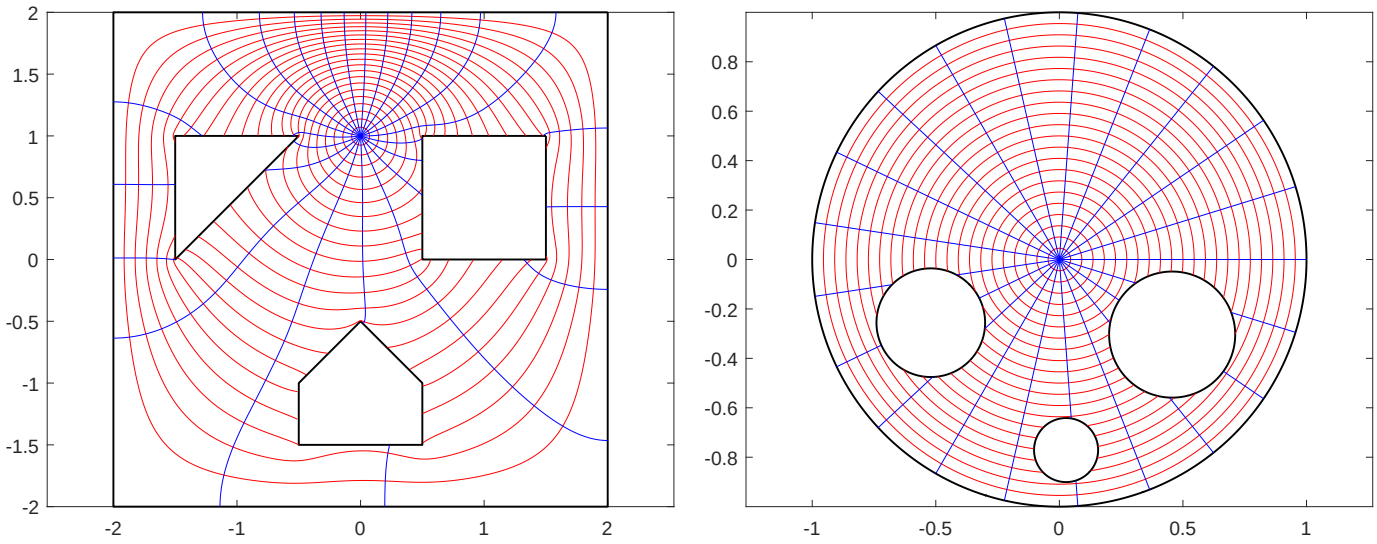
Bounded multiply connected domain $m = 4$

```
plotmap(f,'v','rec',21,21);
```



Bounded multiply connected domain $m = 4$

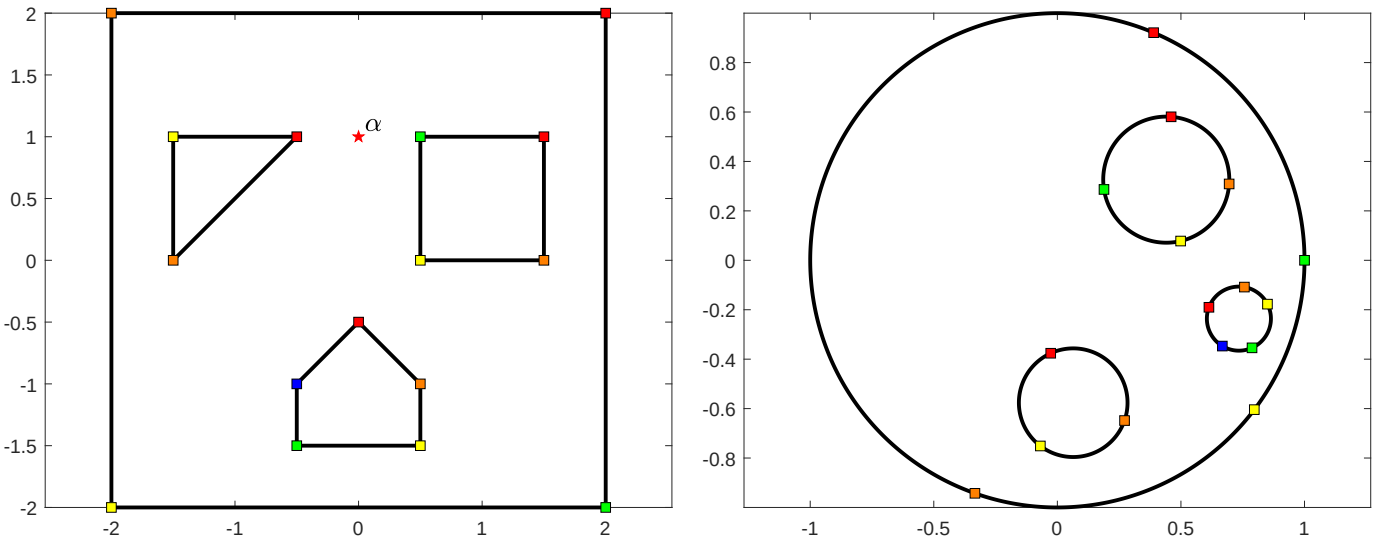
```
plotmap(f,'v','plr',21,21);
```



Bounded multiply connected domain $m = 4$

```
f=plgcirmap(ver,alpha,ver{end}(end));
```

```
plotmap(f);
```



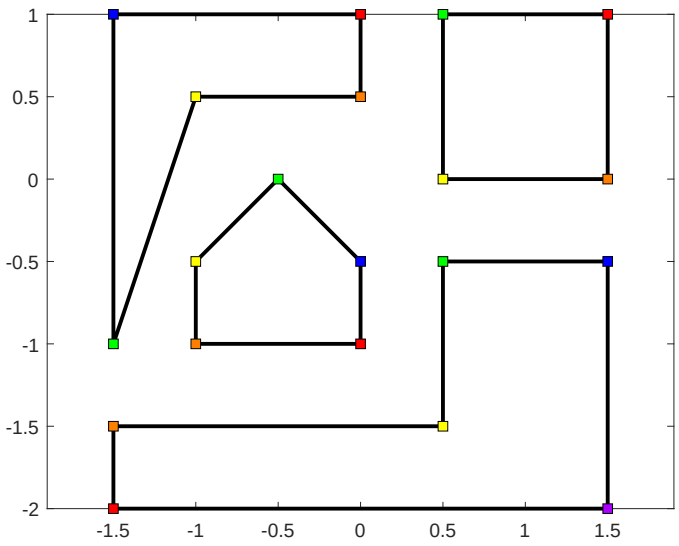
Unbounded multiply connected domain $m = 4$

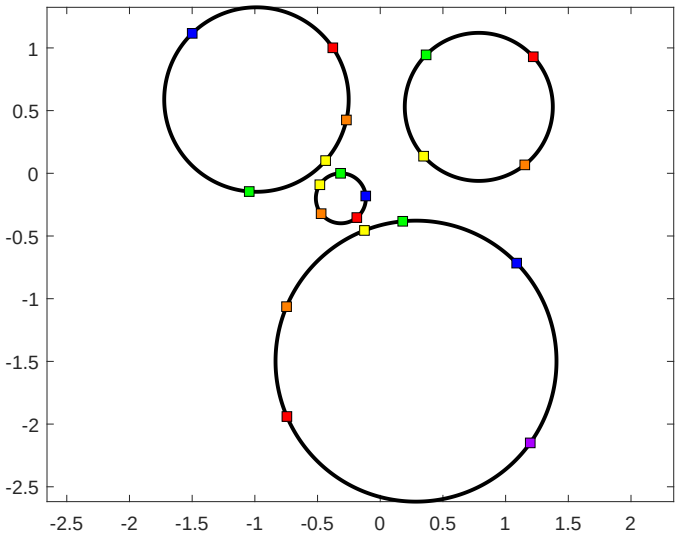
```
clc; clear all;
ver{1}=[ 1.5+1.0i ;  1.5+0.0i ;  0.5+0.0i ; 0.5+1.0i];
ver{2}=[ 0.0+1.0i ;  0.0+0.5i ; -1.0+0.5i ; -1.5-1.0i ; ...
    -1.5+1.0i];
ver{3}=[-0.5-0.0i ;  0.0-0.5i ;  0.0-1.0i ; -1.0-1.0i ; ...
    -1.0-0.5i];
ver{4}=[ 1.5-0.5i ;  1.5-2.0i ; -1.5-2.0i ; -1.5-1.5i ;  ...
    0.5-1.5i ;  0.5-0.5i];
alpha = inf;
```

```
f=plgcirmap(ver,alpha);
```

Unbounded multiply connected domain $m = 4$

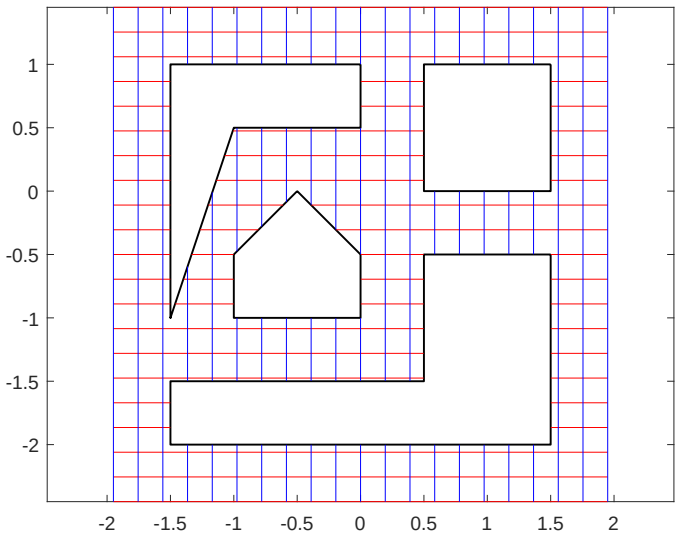
```
plotmap(f);
```

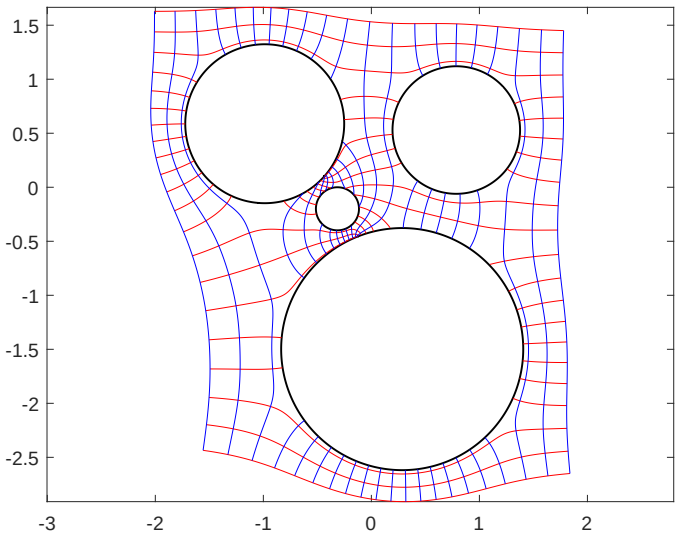




Unbounded multiply connected domain $m = 4$

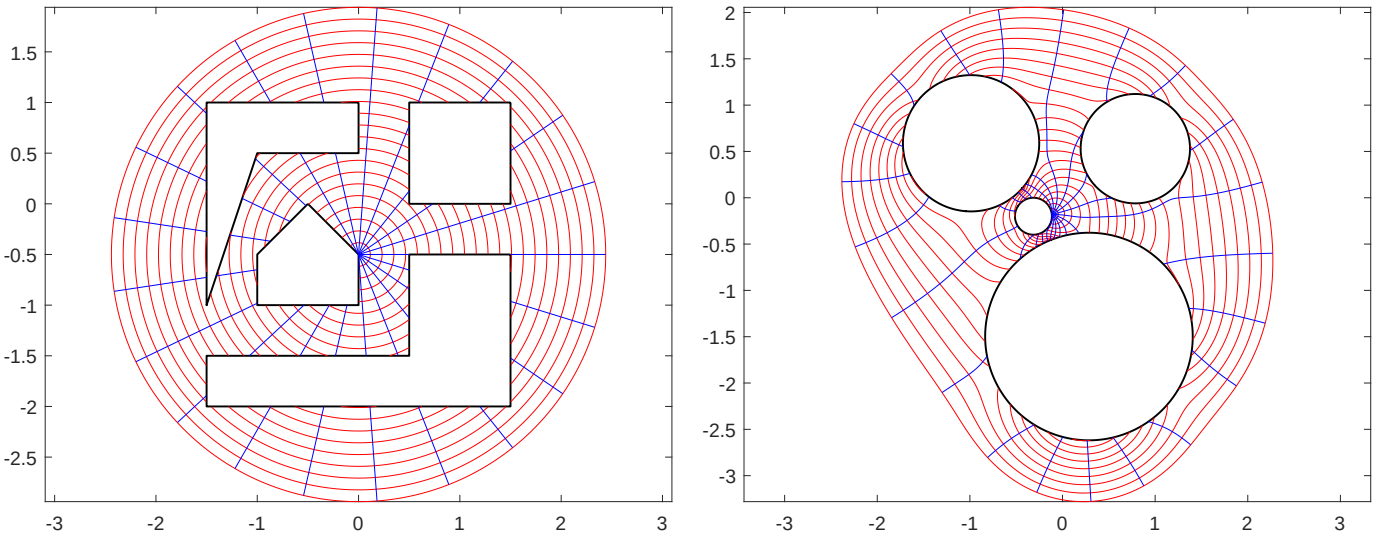
```
plotmap(f,'d','rec',21,21);
```





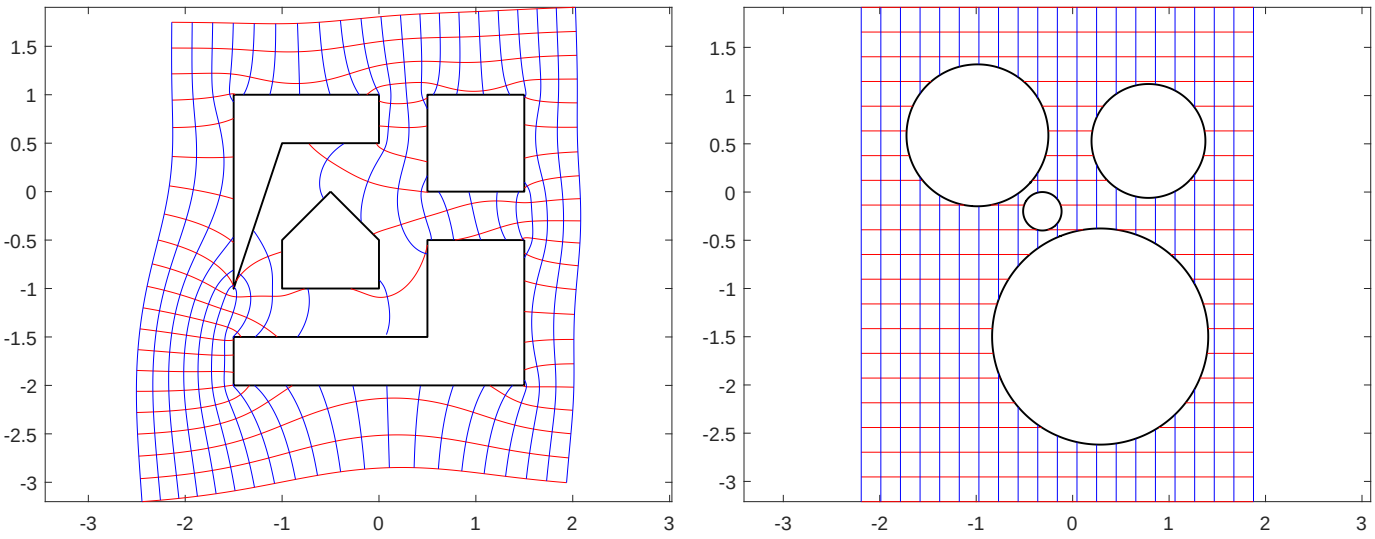
Unbounded multiply connected domain $m = 4$

```
plotmap(f,'d','plr',21,21);
```



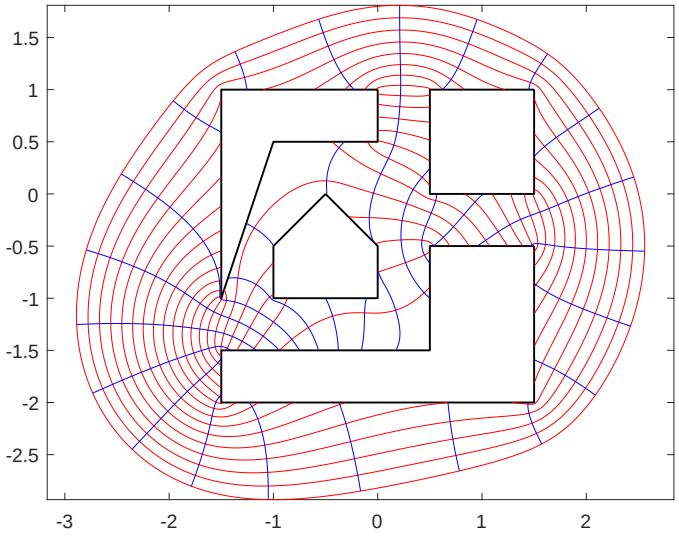
Unbounded multiply connected domain $m = 4$

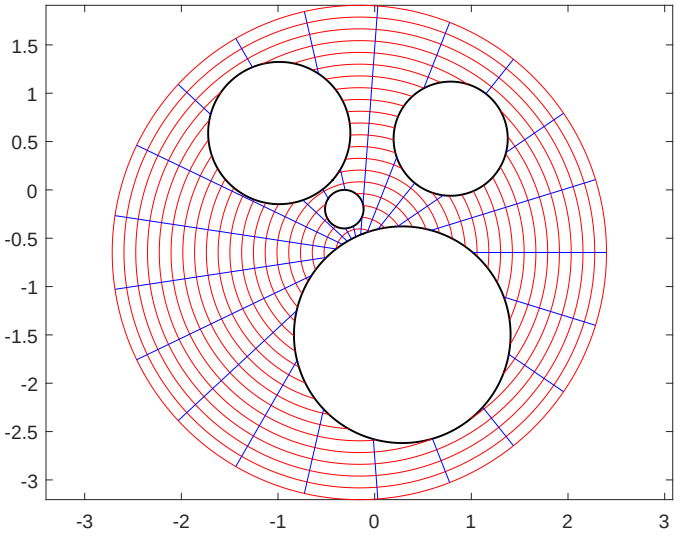
```
plotmap(f,'v','rec',21,21);
```



Unbounded multiply connected domain $m = 4$

```
plotmap(f,'v','plr',21,21);
```

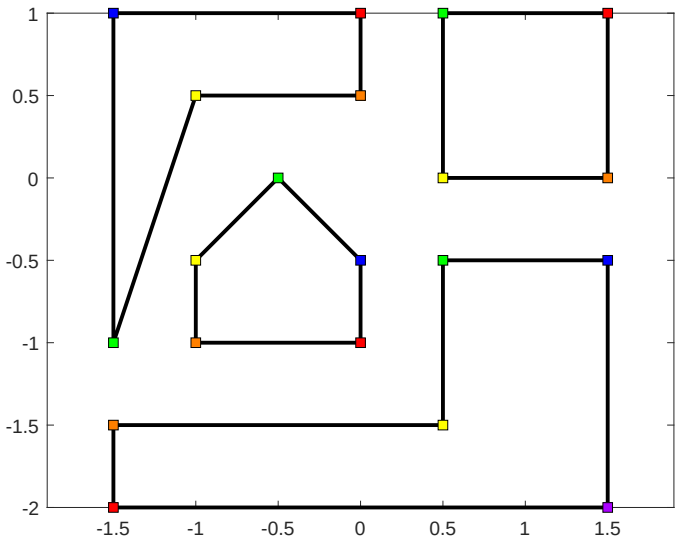


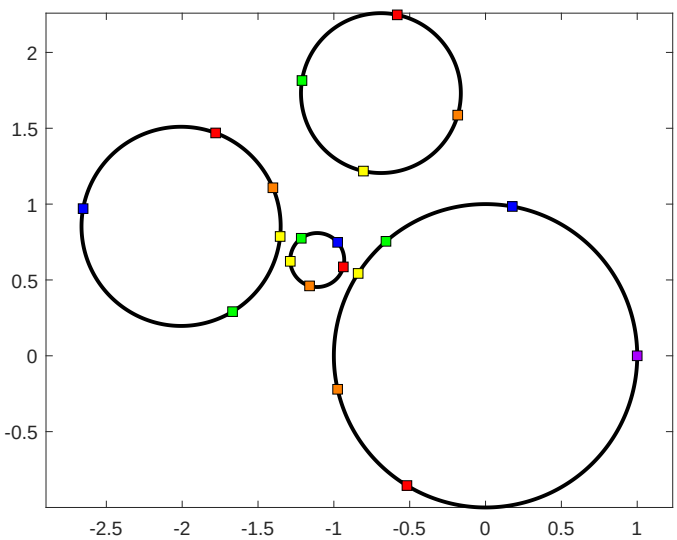


Unbounded multiply connected domain $m = 4$

```
f=plgcirmap(ver,alpha,ver{end}(end));
```

```
plotmap(f);
```





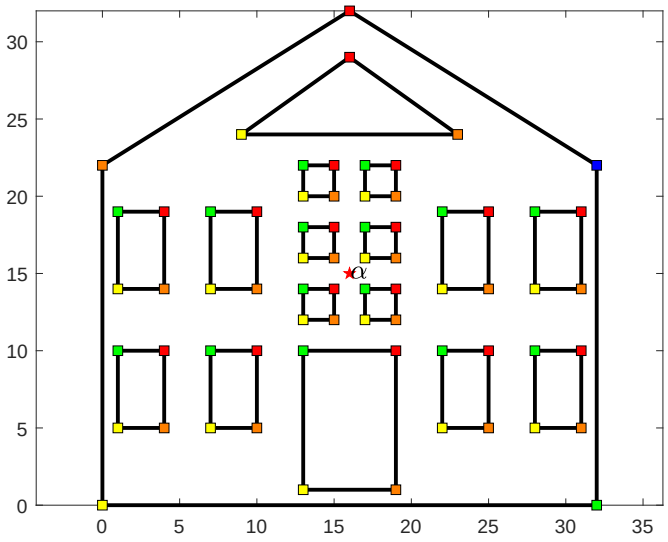
Bounded multiply connected domain $m = 17$

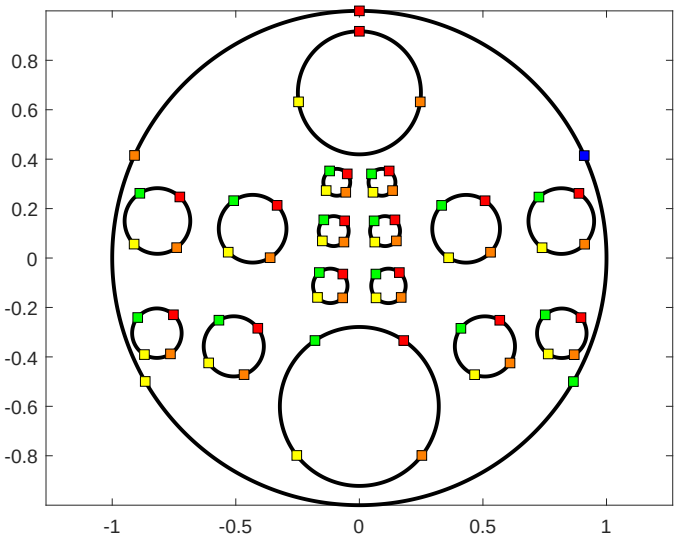
```
ver{1}  = [31+10i ; 31+5i  ; 28+5i  ; 28+10i ];
ver{2}  = [25+10i ; 25+5i  ; 22+5i  ; 22+10i ];
ver{3}  = [19+10i ; 19+1i  ; 13+1i  ; 13+10i ];
ver{4}  = [10+10i ; 10+5i  ;  7+5i  ;  7+10i ];
ver{5}  = [ 4+10i ;  4+5i  ;  1+5i  ;  1+10i ];
ver{6}  = [31+19i ; 31+14i ; 28+14i ; 28+19i ];
ver{7}  = [25+19i ; 25+14i ; 22+14i ; 22+19i ];
ver{8}  = [19+14i ; 19+12i ; 17+12i ; 17+14i ];
ver{9}  = [15+14i ; 15+12i ; 13+12i ; 13+14i ];
ver{10} = [19+18i ; 19+16i ; 17+16i ; 17+18i ];
ver{11} = [15+18i ; 15+16i ; 13+16i ; 13+18i ];
ver{12} = [19+22i ; 19+20i ; 17+20i ; 17+22i ];
ver{13} = [15+22i ; 15+20i ; 13+20i ; 13+22i ];
ver{14} = [10+19i ; 10+14i ;  7+14i ;  7+19i ];
ver{15} = [ 4+19i ;  4+14i ;  1+14i ;  1+19i ];
ver{16} = [16+29i ; 23+24i ;  9+24i ];
ver{17} = [16+32i ;  0+22i ;  0+0i  ; 32+0i  ; 32+22i ];
alpha = 16+15i;
```


Bounded multiply connected domain $m = 17$

```
f=plgcirmap(ver,alpha);
```

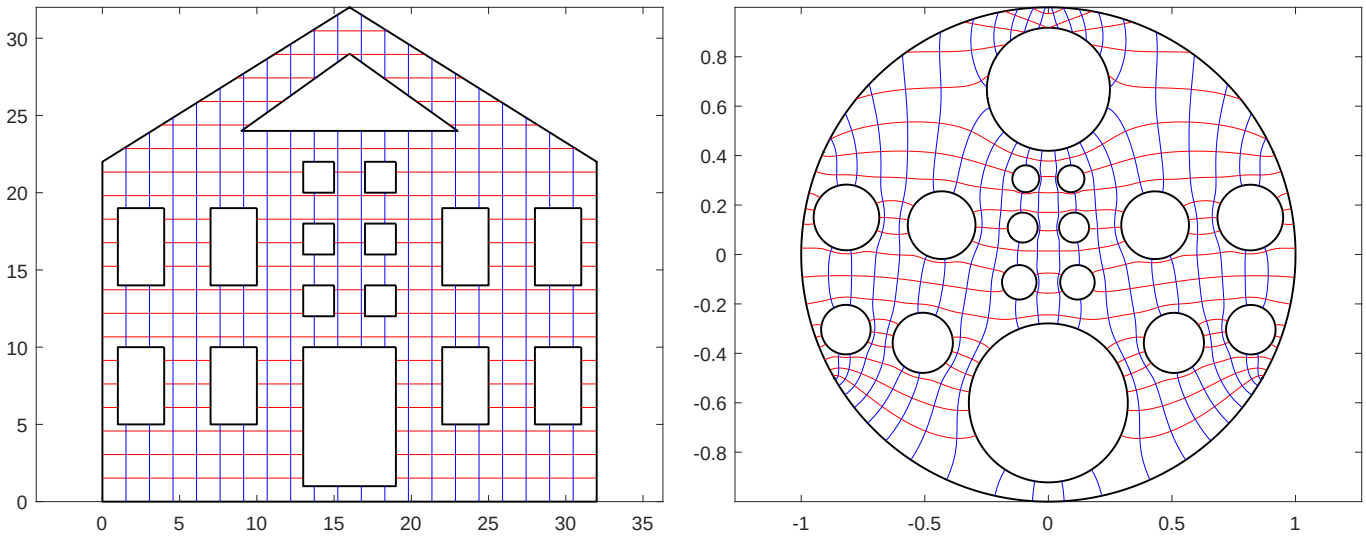
```
plotmap(f);
```





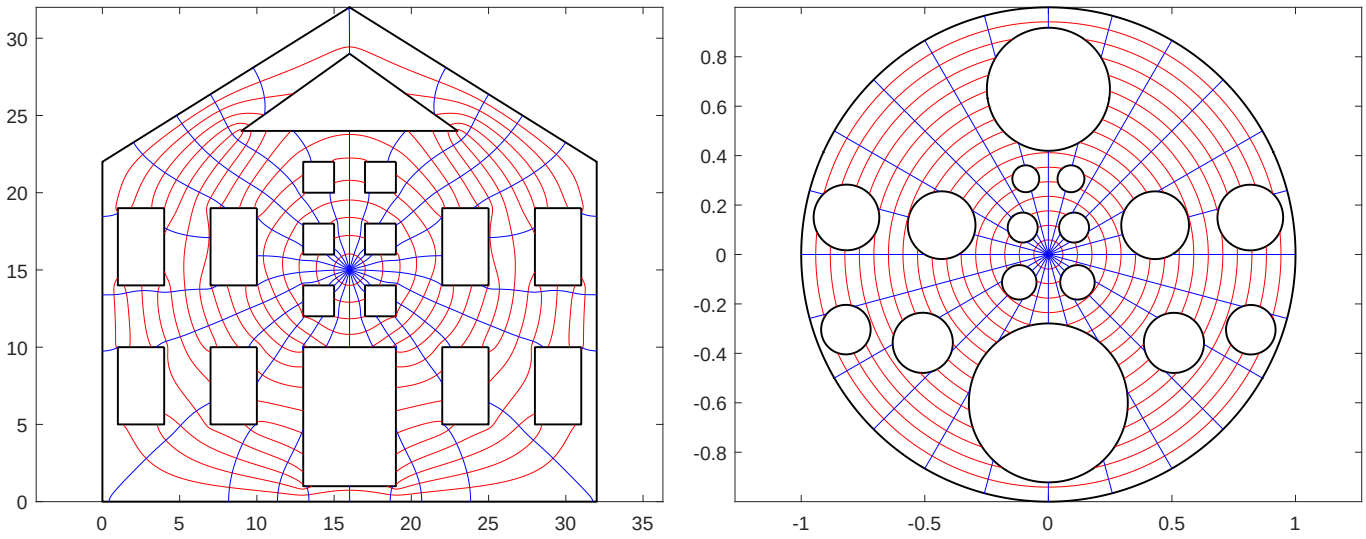
Bounded multiply connected domain $m = 17$

```
plotmap(f,'d','rec');
```



Bounded multiply connected domain $m = 17$

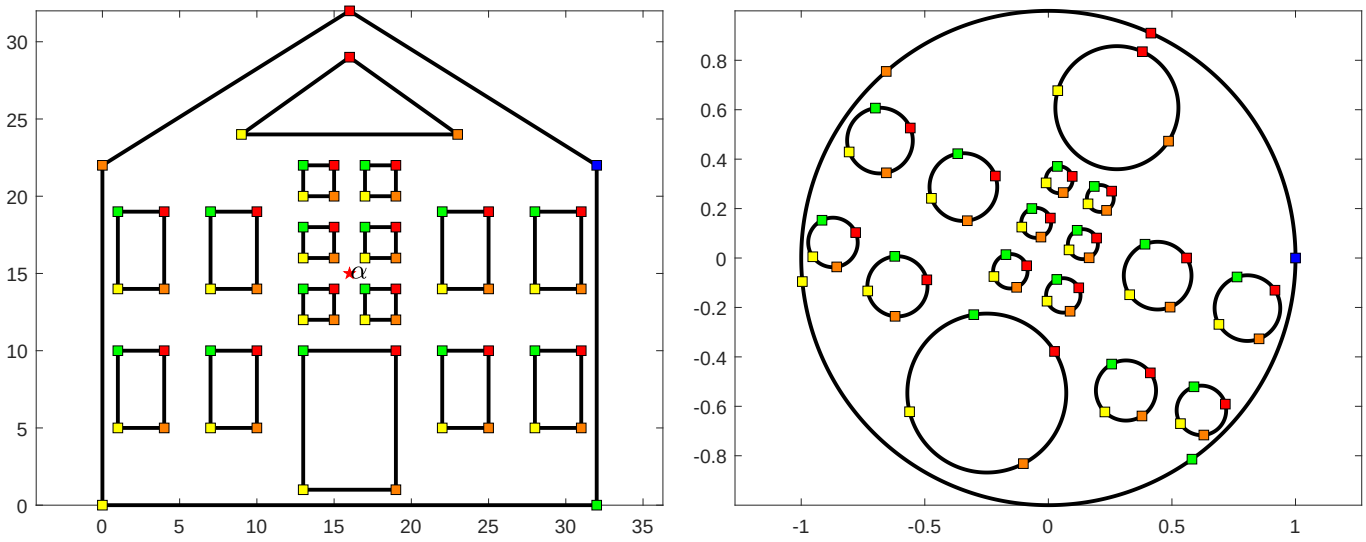
```
plotmap(f,'v','plr');
```



Bounded multiply connected domain $m = 17$

```
f=plgcirmap(ver,alpha,ver{end}(end));
```

```
plotmap(f);
```



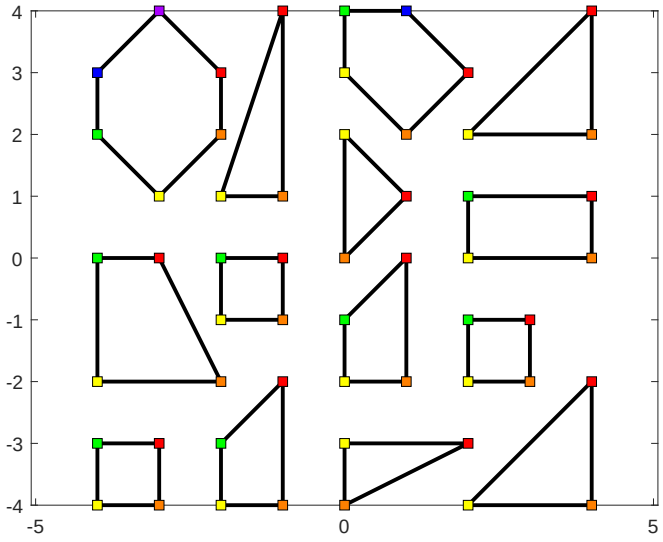
Example 4: Unbounded multiply connected domain $m = 14$

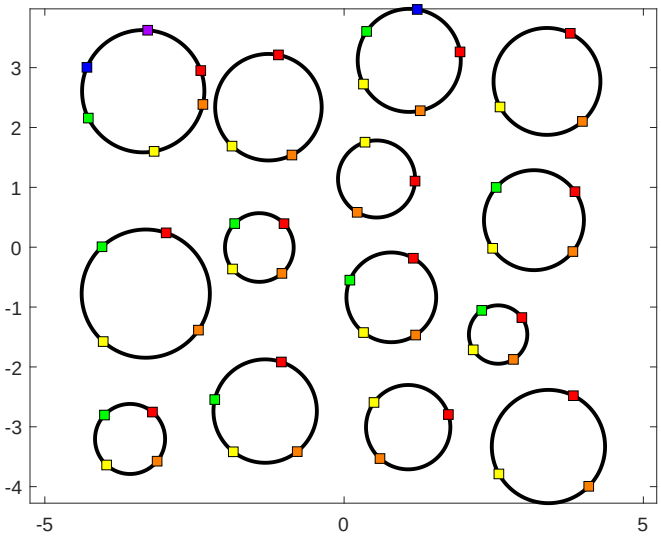
```
ver{1}  = [ 4+4i ; 4+2i ; 2+2i ];
ver{2}  = [ 4+1i ; 4+0i ; 2+0i ; 2+1i ];
ver{3}  = [ 3-1i ; 3-2i ; 2-2i ; 2-1i ];
ver{4}  = [ 4-2i ; 4-4i ; 2-4i ];
ver{5}  = [ 2+3i ; 1+2i ; 0+3i ; 0+4i ; 1+4i ];
ver{6}  = [ 1+1i ; 0+0i ; 0+2i ];
ver{7}  = [ 1+0i ; 1-2i ; 0-2i ; 0-1i ];
ver{8}  = [ 2-3i ; 0-4i ; 0-3i ];
ver{9}  = [-1+4i ;-1+1i ;-2+1i ];
ver{10} = [-1+0i ;-1-1i ;-2-1i ;-2+0i ];
ver{11} = [-1-2i ;-1-4i ;-2-4i ;-2-3i ];
ver{12} = [-2+3i ;-2+2i ;-3+1i ;-4+2i ;-4+3i ;-3+4i ];
ver{13} = [-3+0i ;-2-2i ;-4-2i ;-4+0i ];
ver{14} = [-3-3i ;-3-4i ;-4-4i ;-4-3i ];
alpha = inf;
```

Unbounded multiply connected domain $m = 14$

```
f=plgcirmap(ver,alpha);
```

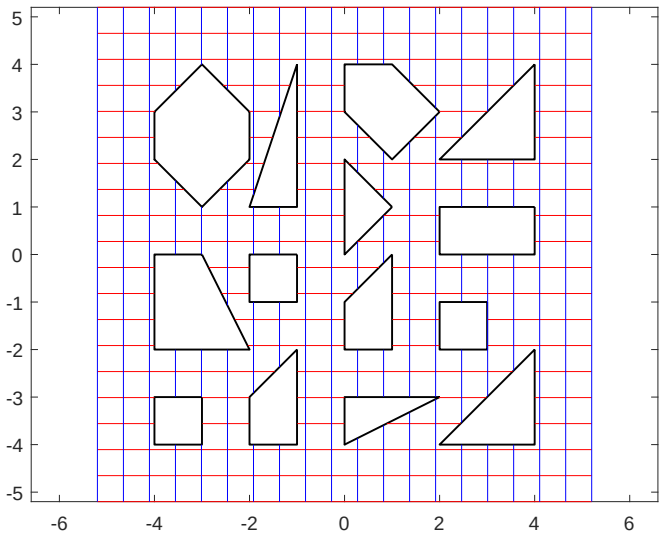
```
plotmap(f);
```

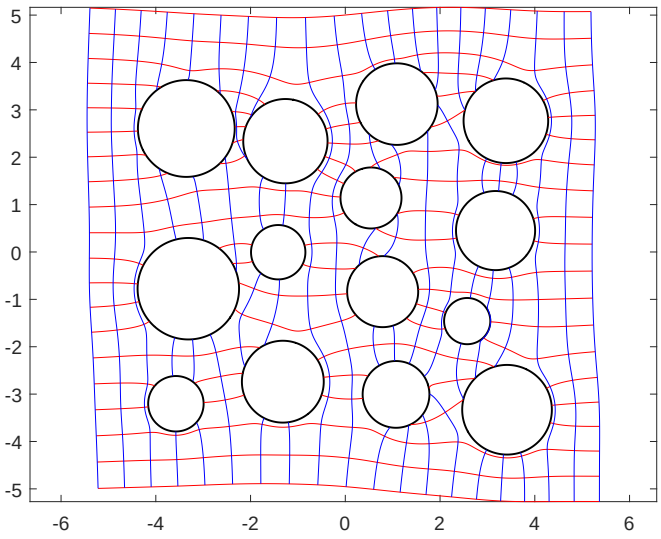




Unbounded multiply connected domain $m = 14$

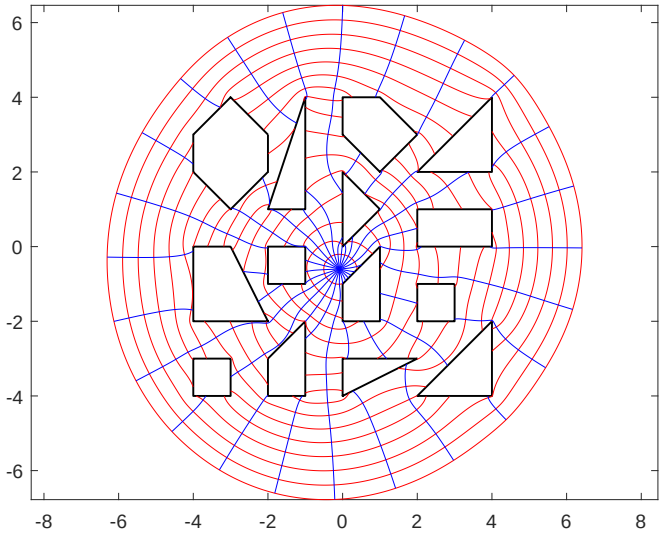
```
plotmap(f,'d','rec');
```

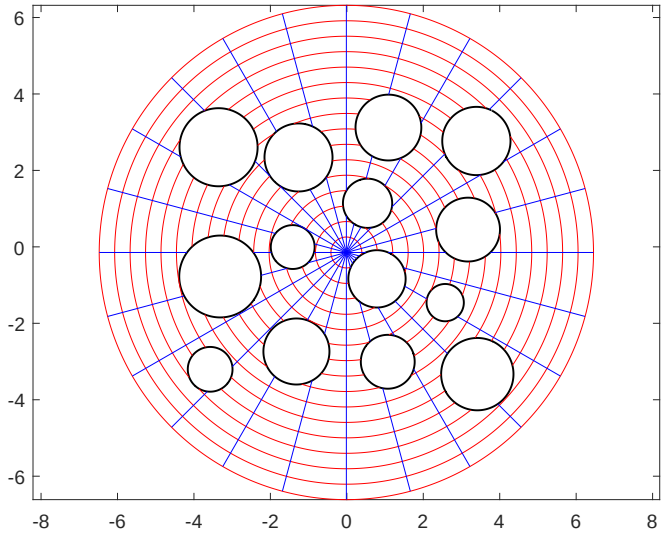




Unbounded multiply connected domain $m = 14$

```
plotmap(f,'v','plr');
```

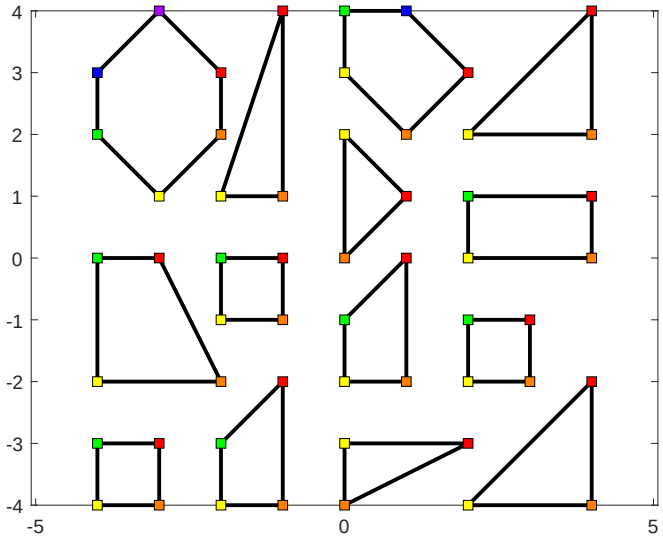


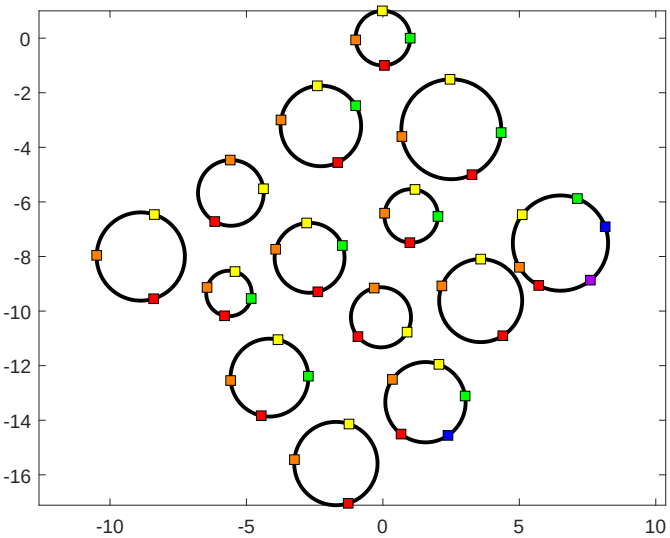


Unbounded multiply connected domain $m = 14$

```
f=plgcirmap(ver,alpha,ver{end}(end));
```

```
plotmap(f);
```





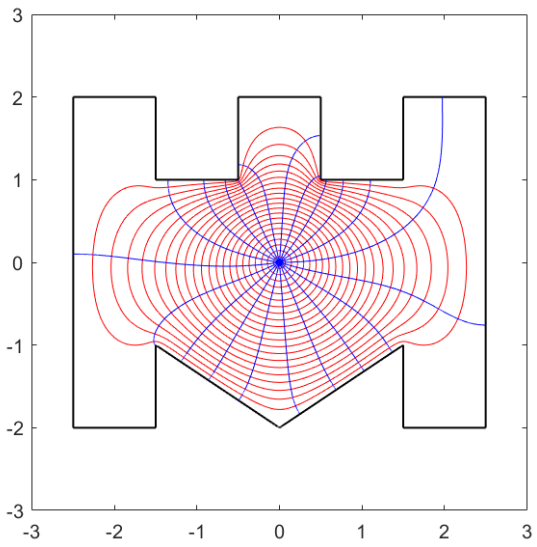
Example 5: Bounded simply connected domain

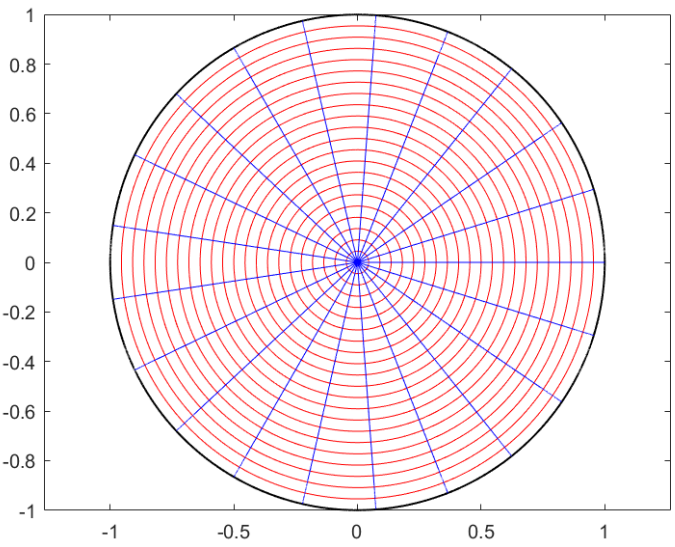
```
ver{1}=[ 2.5+2i ;  1.5+2i ;  1.5+i ;  0.5+i  ; 0.5+2i ;...
        -0.5+2i ; -0.5+i  ; -1.5+i ; -1.5+2i ; -2.5+2i ; ...
        -2.5-2i ; -1.5-2i ; -1.5-i ;  0.0-2i ;...
         1.5-i  ;  1.5-2i ;  2.5-2i];
alpha =  0;
```

```
f=plgcirmap(ver,alpha);
```


Bounded simply connected domain

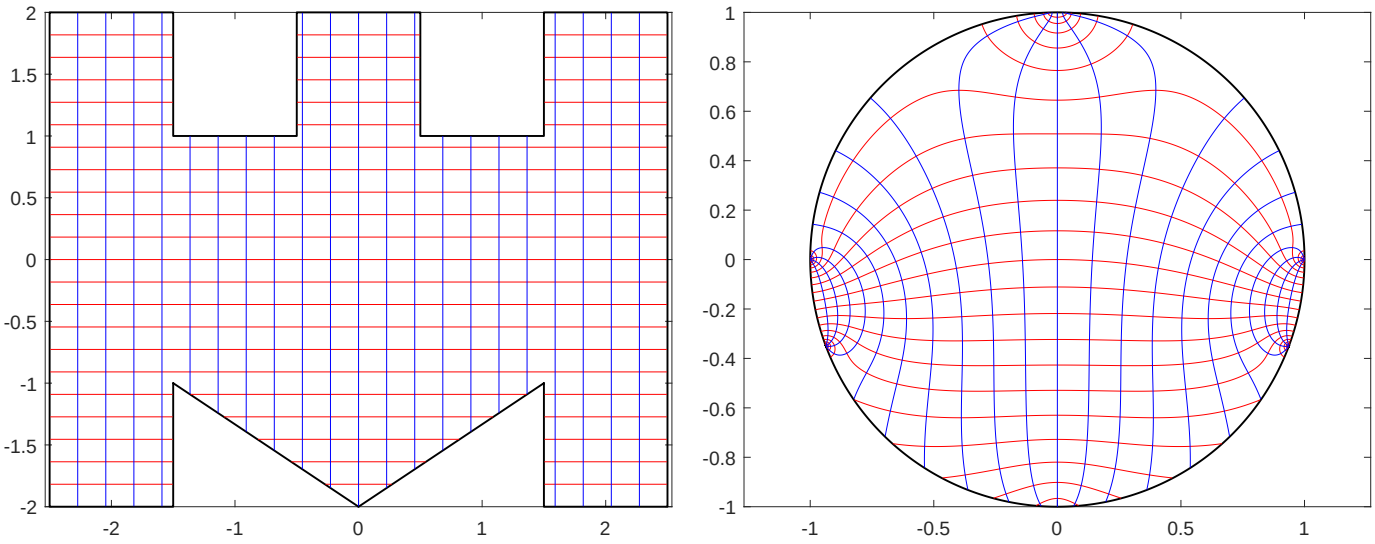
```
plotmap(f,'v','plr',21,21);
```





Bounded simply connected domain

```
plotmap(f,'d','rec',21,21);
```



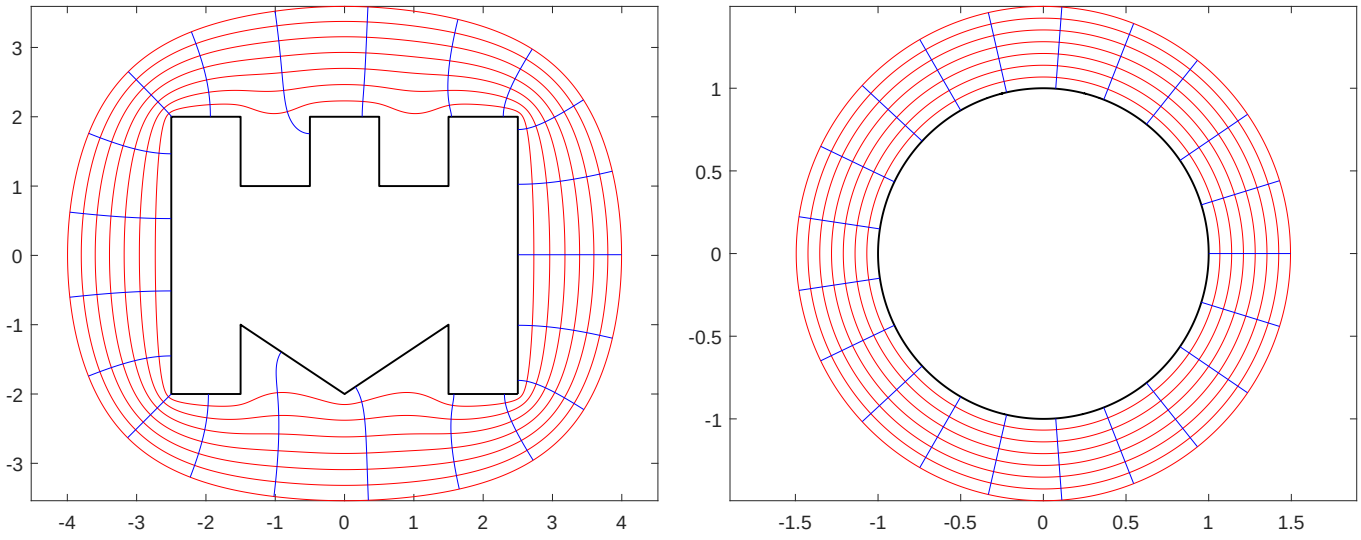
Example 6: Unbounded simply connected domain

```
ver{1}=[ 2.5-2i ; 1.5-2i ; 1.5-i  ; 0.0-2i ; -1.5-i ;...
        -1.5-2i ;-2.5-2i ;-2.5+2i ;-1.5+2i ; -1.5+i ;...
        -0.5+i  ;-0.5+2i ; 0.5+2i ; 0.5+i  ;  1.5+i ;...
         1.5+2i ; 2.5+2i ];
alpha =  inf;
```

```
f=plgcirmap(ver,alpha);
```

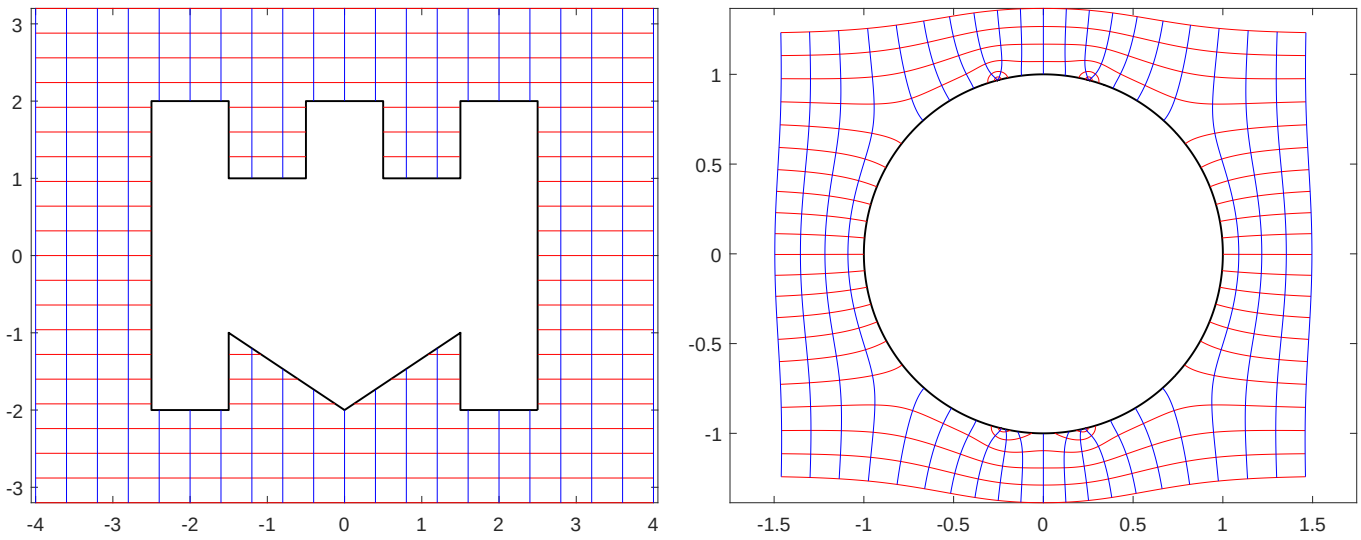
Unbounded simply connected domain

```
plotmap(f,'v','plr',21,21);
```



Unbounded simply connected domain

```
plotmap(f,'d','rec',21,21);
```



Outline

Comparison with Schwarz-Christoffel toolbox

Example 1

```
clc; clear all;
ver{1}=[1i ; -1+1i ; -1-1i ; 1.5-1i ; 1.5 ; 1];
alpha = 0;
```

Using PlgCirMap toolbox:

```
f=plgcirmap(ver,alpha,ver{1}(end));
```

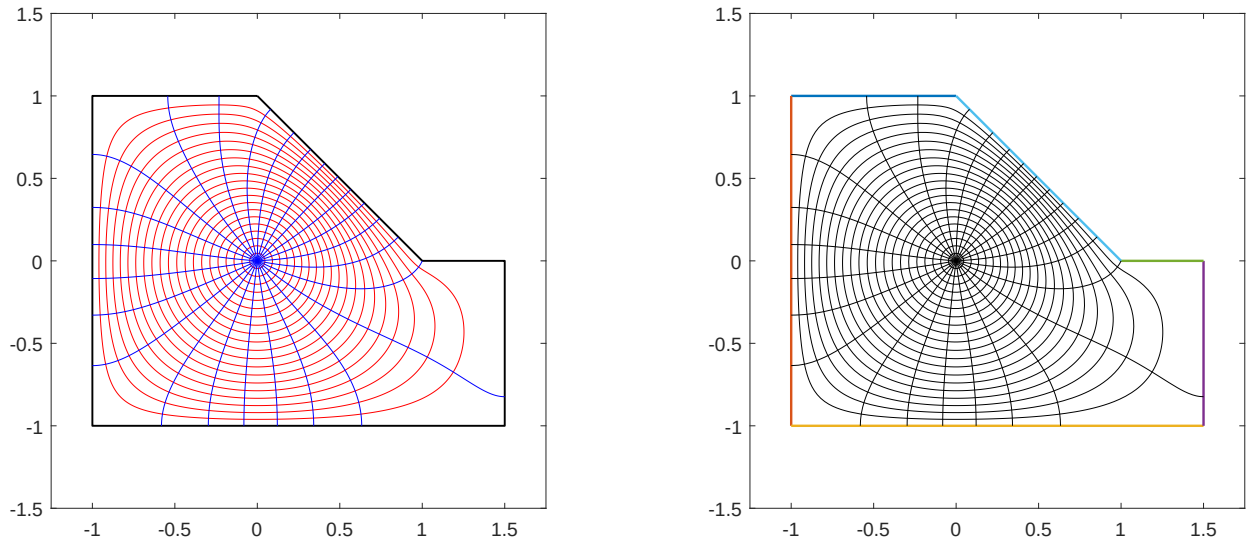
Using SC toolbox:

```
options = scmapopt('Tolerance',1e-14);
p=polygon(ver{1});
fsc = diskmap(p,options);
fsc = center(fsc,alpha);
```

Comparison with Schwarz-Christoffel toolbox

Example 1

```
plotmap(f,'v','plr',20,25);
figure
plot(fsc,20,25)
```



Comparison with Schwarz-Christoffel toolbox

Example 1

```
prevertsc = get(fsc,'prevert');
prevertpc = f.imgver{:};
error_prevert = norm(prevertsc-prevertpc,inf)
```

```
9.1372e-13
```

```
zz  = 0.6.*exp(i.*linspace(0,2*pi,1000));
wzzpc = evalu(f,zz,'d');
wzzsc = eval(inv(fsc),zz);
error_map = norm(wzzpc-wzzsc,inf)
```

```
2.6342e-13
```

```
error_pc = norm(zz-evalu(f,wzzpc,'v'),inf)
```

```
8.8130e-14
```

```
error_sc = norm(zz-fsc(wzzsc),inf)
```

```
7.5906e-15
```

Comparison with Schwarz-Christoffel toolbox

Example 2

```
clc; clear all;
ver{1} = [ 3+0i; 2+i; 2+2i; 1+2i; 0+3i; -1+2i; -2+2i;...
          -2+1i; -3+0i;-2-1i; -2-2i; -1-2i; 0-3i; 1-2i;...
                      2-2i; 2-1i];
alpha =  1i;
```

Using PlgCirMap toolbox:

```
f=plgcirmap(ver,alpha,ver{1}(end));
```

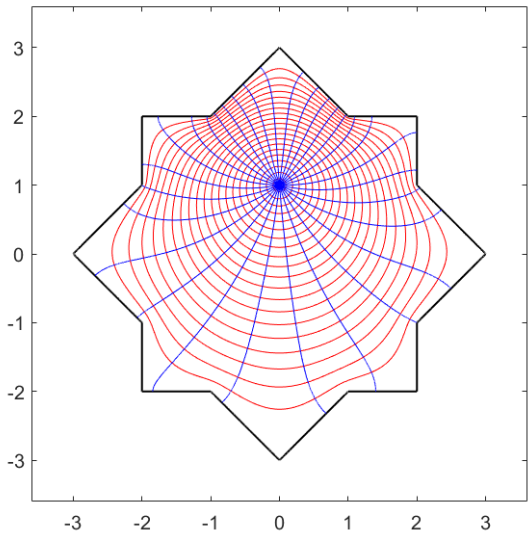
Using SC toolbox:

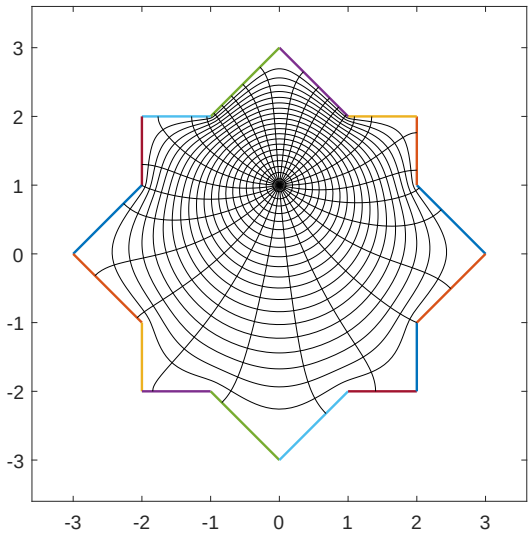
```
options = scmapopt('Tolerance',1e-14);
p=polygon(ver{1});
fsc = diskmap(p,options);
fsc = center(fsc,alpha);
```


Comparison with Schwarz-Christoffel toolbox

Example 2

```
plotmap(f,'v','plr',20,30);
figure
plot(fsc,20,30)
```





Comparison with Schwarz-Christoffel toolbox

Example 2

```
prevertsc = get(fsc,'prevert');
prevertpc = f.imgver{:};
error_prevert = norm(prevertsc-prevertpc,inf)
```

```
1.1367e-12
```

```
zz  = 1.99.*exp(i.*linspace(0,2*pi,1000));
wzzpc = evalu(f,zz,'d');
wzzsc = eval(inv(fsc),zz);
error_map = norm(wzzpc-wzzsc,inf)
```

```
8.4306e-13
```

```
error_pc = norm(zz-evalu(f,wzzpc,'v'),inf)
```

```
1.5750e-13
```

```
error_sc = norm(zz-fsc(wzzsc),inf)
```

```
2.3735e-14
```


Comparison with Schwarz-Christoffel toolbox

Example 3

```
clc; clear all;
ver{1}=[ 2.5+2i ;  1.5+2i ;  1.5+i ; -1.5+i ; -1.5+2i ;...
        -2.5+2i ; -2.5-2i ; -1.5-2i ; -1.5-i ; 1.5-i  ;...
                  1.5-2i ;  2.5-2i];
alpha =  1;
```

Using PlgCirMap toolbox:

```
f=plgcirmap(ver,alpha,ver{1}(end));
```

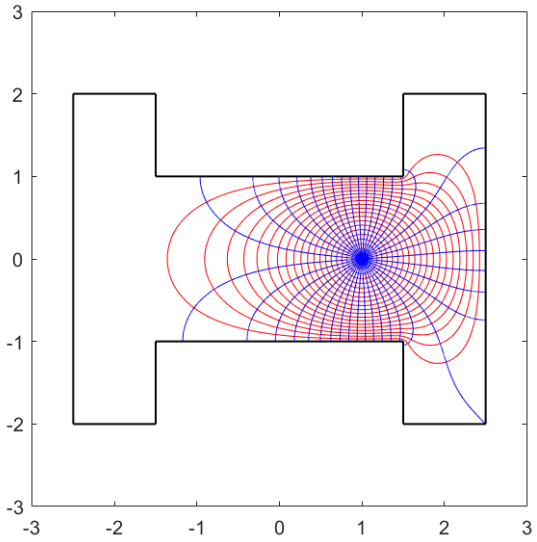
Using SC toolbox:

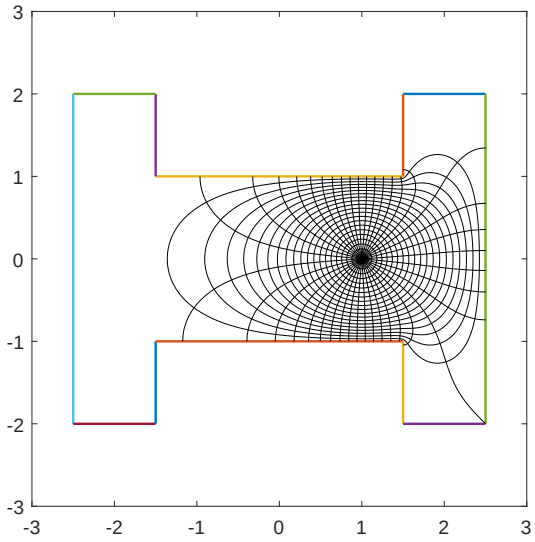
```
options = scmapopt('Tolerance',1e-14);
p=polygon(ver{1});
fsc = diskmap(p,options);
fsc = center(fsc,alpha);
```

Comparison with Schwarz-Christoffel toolbox

Example 3

```
plotmap(f,'v','plr',20,40);
figure
plot(fsc,20,40)
```





Comparison with Schwarz-Christoffel toolbox

Example 3

```
prevertsc = get(fsc,'prevert');
prevertpc = f.imgver{:};
error_prevert = norm(prevertsc-prevertpc,inf)
```

```
4.0014e-10
```

```
zz  = 0.99.*exp(i.*linspace(0,2*pi,1000));
wzzpc = evalu(f,zz,'d');
wzzsc = eval(inv(fsc),zz);
error_map = norm(wzzpc-wzzsc,inf)
```

```
6.3285e-10
```

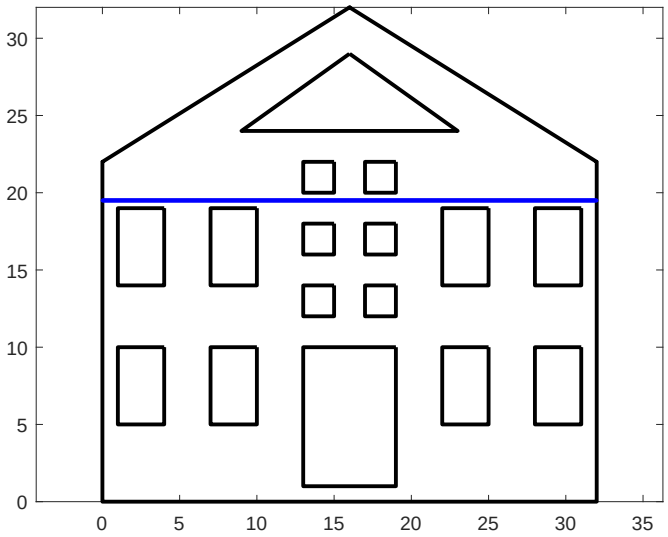
```
error_pc = norm(zz-evalu(f,wzzpc,'v'),inf)
```

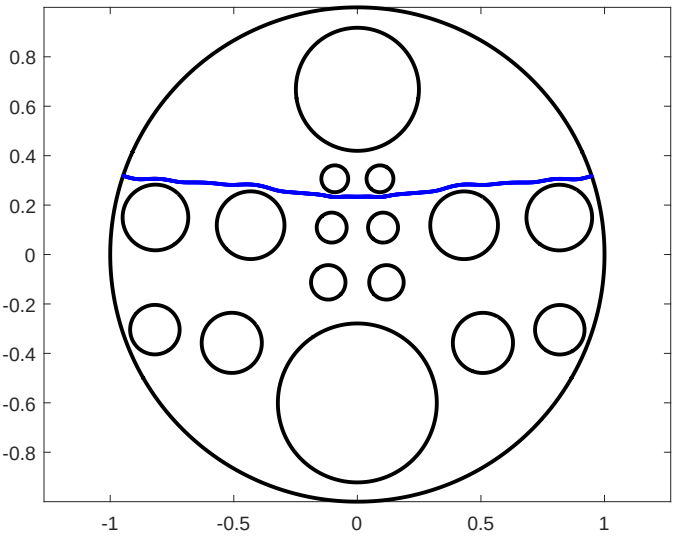
```
3.2666e-12
```

```
error_sc = norm(zz-fsc(wzzsc),inf)
```

```
2.0991e-12
```

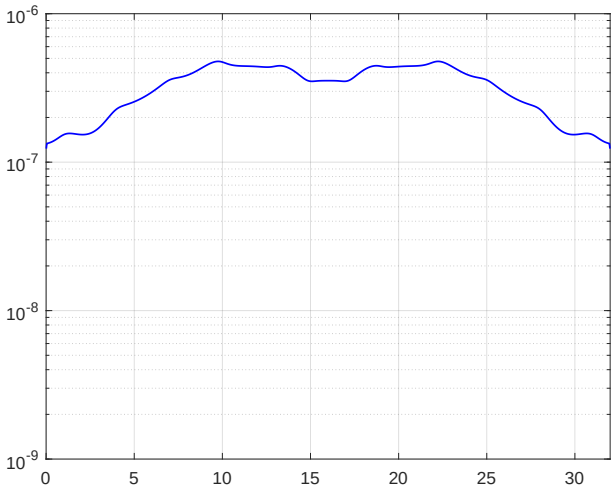
Bounded multiply connected domains



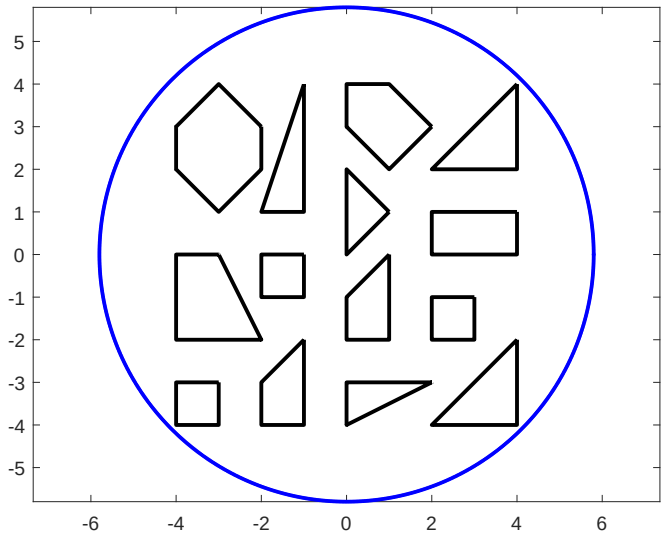


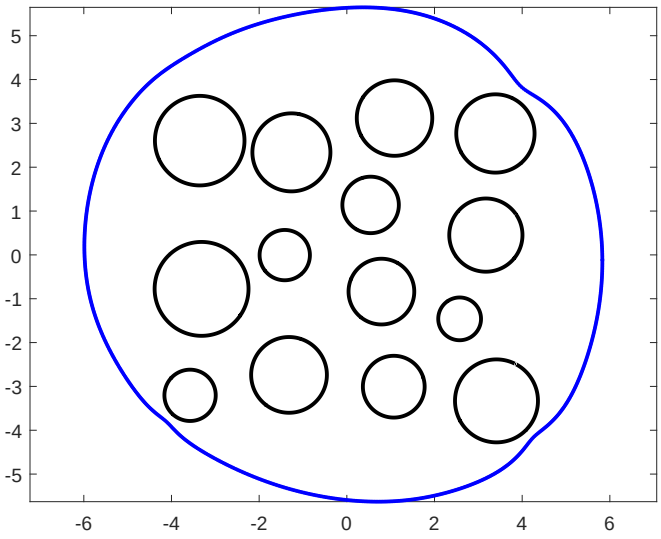
Bounded multiply connected domains

```
zz    =  linspace(0.01,31.99,1000)+19.5i;
wzz   =  evalu(f,zz,'d');
zzi   =  evalu(f,wzz,'v');
Error = abs(zz-zzi);
figure
semilogy(real(zz),Error,'b','LineWidth',1);
grid on
```



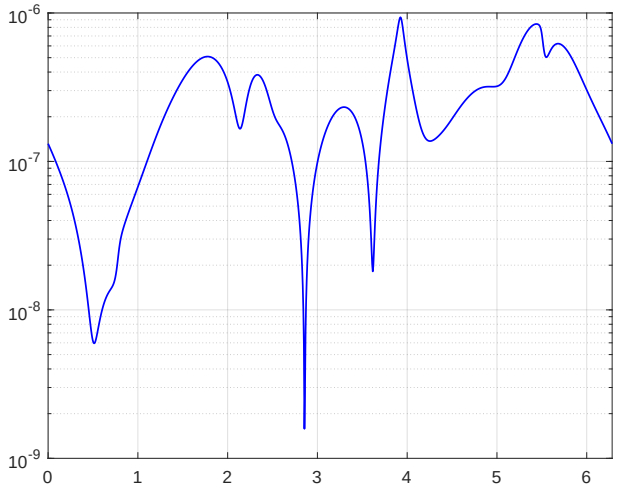
Unbounded multiply connected domains





Unbounded multiply connected domains

```
tt      =  linspace(0,2*pi,1000);
zz      =  5.8.*exp(i.*tt);
wzz     =  evalu(f,zz,'d');
zzi     =  evalu(f,wzz,'v');
Error = abs(zz-zzi);
figure
semilogy(real(zz),Error,'b','LineWidth',1);
grid on
```



Outline

Point vortex trajectories in polygonal domains

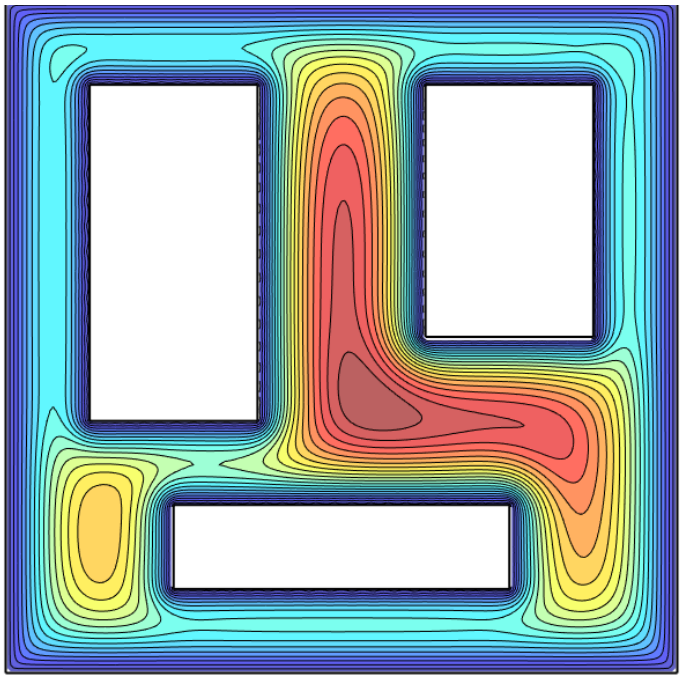
Let $w = f(z)$ be the conformal mapping from a multiply connected polygonal domain G on a circular domain D and $z = g(w)$ be its inverse.
The trajectories of a point vortex in the polygonal domain G can be written as the contours of the so-called Kirchhoff-Routh path function given explicitly in terms of the Schottky-Klein prime function as[7]

$$H(z,\overline{z}) = -\frac{1}{8\pi}\log\left|\frac{\hat{\omega}(w,w)\overline{\hat{\omega}}(1/w,1/w)}{w^2\omega(w,1/\overline{w})\overline{\omega}(1/w,\overline{w})g'(w)^2}\right|,$$

where $w = f(z)$ and $\omega(w_1,w_2) = (w_1 - w_2)\hat{\omega}(w_1,w_2)$.

[7] D.G. Crowdy and J.S. Marshall, Analytical formulae for the Kirchhoff-Routh path function in multiply connected domains. Proc. R. Soc. Lond. A 461 (2005) 2477–2501.

Point vortex trajectories in polygonal domain of connectivity 4



THANK YOU