2D Ideal Uniform Fluid Flow Around an Obstacle

December 13, 2018

Project
Methods of Mathematical Physics
Teacher: Dr. Sheehan Olver

Tianyi Pu
CID: 01553448
Due Date: 14/12/2018

1 Introduction

The LectureNote#12[1] introduces a method to find a 2D ideal uniform fluid flow around a segment or multiple segments within the real line. In this project, I am going to find 2D ideal uniform fluid flows around a simply connected bounded obstacle.

A 2D flow $v = v(x, y)$ denotes the stream function of flow where $(x, y) \in \mathbb{R}^2$. An ideal fluid flow v satisfies Laplace equation:

$$v_{xx} + v_{yy} = 0.$$

An obstacle is a simply connected bounded open set of $\mathbb{R}^2$ denoted by D. Without loss of generality, we assume that $0 \in D$. v is defined on $\mathbb{R}^2\backslash D$ while $v(\partial D) = \{0\}$. Thus we can model an ideal fluid flow around obstacle(s) D as

$$v_{xx} + v_{yy} = 0, \quad (x, y) \in \mathbb{R}^2\backslash\bar{D},$$

$$v(x, y) = 0, \quad (x, y) \in \partial D.$$

Furthermore, the flow is uniform, meaning that v tends to a uniform flow v_0 as $|(x, y)| \to \infty$. Without loss of generality, we assume that $v_0(x, y) = y$ and then

$$v(x, y) \sim y \quad \text{as} \quad |(x, y)| \to \infty.$$

Since both real part and imaginary part of a holomorphic complex function automatically satisfy the Laplace equation, we suppose that v is the imaginary part of an unknown holomorphic complex function $\phi(z)$ defined on $\mathbb{C}\backslash D$ which satisfies the following conditions:

$$\phi \to z + c \quad \text{as} \quad z \to \infty$$

$$\mathrm{Im}(\phi(\partial D)) = \{0\}$$

Let $\psi = \phi - z$, then ψ satisfies:

$$\psi \to c \quad \text{as} \quad z \to \infty$$

$$\mathrm{Im}(\psi(z)) = -\mathrm{Im}(z) \quad \text{for} \quad z \in \partial D.$$

As long as ψ is known, we obtain ϕ and take the imaginary part of ϕ represented by x and y to get $v(x, y)$, where $x = \mathrm{Re}(z)$ and $y = \mathrm{Im}(z)$.

[1] http://nbviewer.jupyter.org/github/dlfivefifty/M3M6LectureNotes/blob/master/Lecture%2012.ipynb

In section 2, I will show the main procedure to solve this problem. In section 3, I will solve some problems with D of specific shapes. In section 4, I will give a convergent method to solve problems with general D. In section 5, I will give an idea on how to solve problems with multiple obstacles.

All the codes are run under Mathematica 9.0 or Matlab R2016a. Since the code styles in Mathematica and Matlab are much different, I will not tell which environment the codes are in. Note that Matlab codes have to run within a toolbox[2].

2 Main procedure

Step 1: Map $\mathbb{C}\backslash\bar{D}$ onto the interior of a simply connected open set

Let $\psi_1(z) = \psi(1/z)$ for $z \in D_1$ where $D_1 = \{1/z : z \in \mathbb{C}\backslash\bar{D}\}$ is a bounded, simply connected open subset of $\mathbb{C}$. Since ∞ is a removable singularity of $\psi(z)$, define $\psi_1(0) = c$ and $\psi_1(z)$ is holomorphic on D_1 with

$$\mathrm{Im}(\psi_1(z)) = -\mathrm{Im}(1/z) \quad \text{for} \quad z \in \partial D_1.$$

Step 2: Map D_1 onto the unit disk

By Riemann mapping theorem in complex analysis, since D_1 is a simply connected open subset of $\mathbb{C}$, there exists a biholomorphic function f from D_1 onto the open unit disk $D_2 = \{z \in \mathbb{C} : |z| < 1\}$. By the boundary correspondence theorem in complex analysis, f is also a bijection from ∂D_1 to ∂D_2. By the following lemma, we can assume that $f(0) = 0$.

Lemma 1 *There exists a holomorphic automorphism g of D_2 that maps $a \in D_2$ to 0.*

Proof: Let $g(z) = \frac{z-a}{1-\bar{a}z}$. □

Let $\psi_2(z) = \psi_1(f^{-1}(z))$ for $z \in D_2$, then $\psi_2(0) = \psi_1(0) = c$ and

$$\mathrm{Im}(\psi_2(z)) = \mathrm{Im}(\psi_1(f^{-1}(z))) = -\mathrm{Im}(1/f^{-1}(z)) \quad \text{for} \quad |z| = 1.$$

Step 3: Solve a Dirichlet problem and get $\mathrm{Im}(\psi_2(z))$ on D_2

As $\psi_2(z)$ is holomorphic on D_2, it imaginary part denoted by $u(x, y)$ is harmonic where $x = \mathrm{Re}(z)$ and $y = \mathrm{Im}(z)$. Thus we get a Dirichlet problem:

$$\Delta u = 0 \quad \text{in} \quad B(0,1)$$

$$u(x, y) = -\mathrm{Im}(1/f^{-1}(x + iy)) \quad \text{for} \quad (x, y) \in \partial B(0, 1).$$

By Poisson's formula, u is given by

$$u = -\frac{1-r^2}{2\pi} \int_0^{2\pi} \frac{\mathrm{Im}(1/f^{-1}(e^{i\varphi}))}{1 - 2r\cos(\theta - \varphi) + r^2} \mathrm{d}\varphi$$

where $r = |x + iy|$ and $\theta = \arg(x + iy)$.

[2] https://github.com/tobydriscoll/sc-toolbox

Step 4: Reverse all operations to get $\mathrm{Im}(\phi)$

We have obtained u, where $u = \mathrm{Im}(\psi_2(z)) = \mathrm{Im}(\psi_1(f^{-1}(z))) = \mathrm{Im}(\psi(1/f^{-1}(z))) = \mathrm{Im}(\phi(1/f^{-1}(z))) - \mathrm{Im}(1/f^{-1}(z)))$. Thus $\mathrm{Im}(\phi(z)) = \mathrm{Im}(z) + \mathrm{Im}(\psi(z)) = y + \mathrm{Im}(\psi_1(1/z)) = y + \mathrm{Im}(\psi_2(f(1/z))) = y + u(f(1/z))$. In conclusion,

$$\mathrm{Im}(\phi(z)) = y - \frac{1-|f(1/z)|^2}{2\pi}\int_0^{2\pi} \frac{\mathrm{Im}(1/f^{-1}(e^{i\varphi}))}{1-2|f(1/z)|\cos(\arg(f(1/z))-\varphi)+|f(1/z)|^2}\mathrm{d}\varphi. \quad (1)$$

3 Applications of the method

From (1), we know that the key to the method is f, which comes from Riemann mapping theorem, an existence theorem. However, finding f is difficult in general cases, thus this method is limited to a specific class of D. In this section, I will show some cases where f is solvable, and find the corresponding flow.

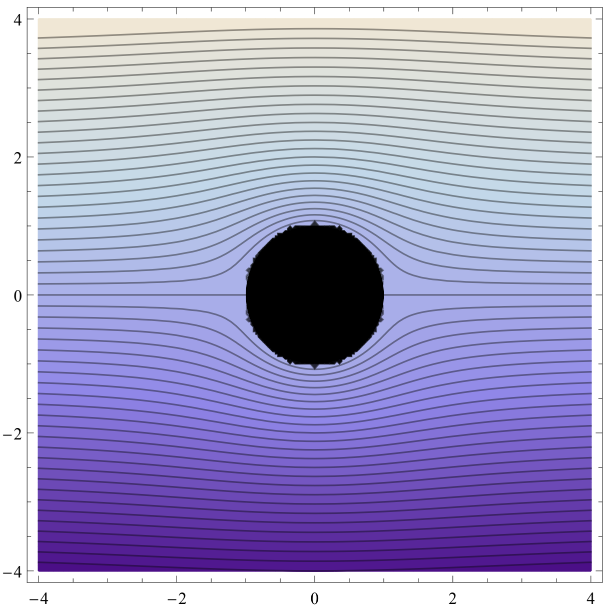
3.1 D is the unit disk

When D is the unit disk, $D_1 = D$ and thus $f(z) = z$.

$$\mathrm{Im}(\phi(z)) = y - \frac{1}{2\pi}\int_0^{2\pi} \frac{(1-|z|^2)\sin\varphi}{|z|^2-2|z|\cos(\arg(z)+\varphi)+1}\mathrm{d}\varphi$$

We can check this result:

```
Show[
  ContourPlot[
    y + (x^2 + y^2 - 1)/(2 Pi)*NIntegrate[
      Sin[phi]/(x^2 + y^2 - 2 Sqrt[x^2 + y^2] Cos[Arg[x + y I] + phi] + 1), {phi,
         0, 2 Pi}],
    {x, -4, 4}, {y, -4, 4}, Contours -> 50],
  Graphics[Disk[{0, 0}, 1]]]
```

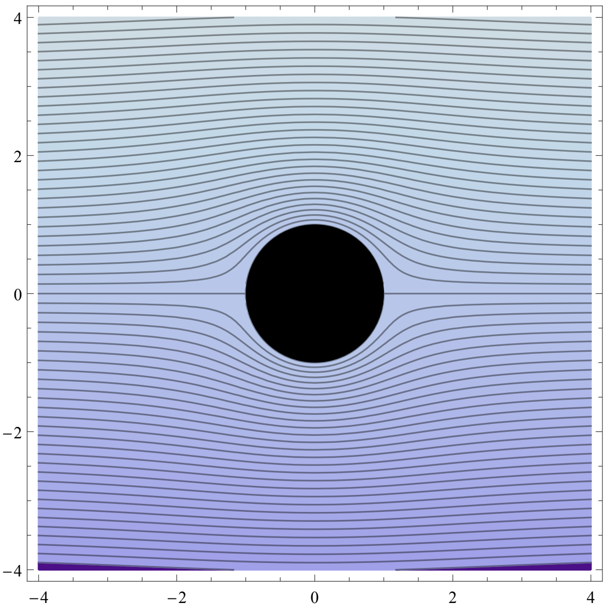


Also we can directly get the explicit expression of the integral:

```
In[1]:= Integrate[(1 - r^2)/(2 Pi)*Sin[phi]/(r^2 - 2 r Cos[theta + phi] + 1), {phi,
    0, 2 Pi}, Assumptions -> {r > 1, 0 < theta < Pi}]
Out[1]= Sin[theta]/r
In[2]:= Integrate[(1 - r^2)/(2 Pi)*Sin[phi]/(r^2 - 2 r Cos[theta + phi] + 1), {phi,
    0, 2 Pi}, Assumptions -> {r > 1, Pi < theta < 2 Pi}]
Out[2]= Sin[theta]/r
```

Thus $\mathrm{Im}(\phi(z)) = y - \frac{\sin\theta}{r} = y - \frac{y}{x^2+y^2}$. Check this result:

```
Show[
  ContourPlot[y - y/(x^2 + y^2), {x, -4, 4}, {y, -4, 4}, Contours -> 100],
  Graphics[Disk[{0, 0}, 1]]]
```



3.2 D is an inverse simple polygon

A simple polygon of $\mathbb{C}$ is a simply connected open subset of $\mathbb{C}$ whose boundary is a chain of line segments of $\mathbb{C}$. For a simple polygon F, its inverse is defined by $F^{-1} = \{1/z : z \in \mathbb{C}\backslash\bar{F}\}$. Thus D is an inverse simple polygon means that D_1 is a simple polygon.

Schwartz-Christoffel formula[3] gives a biholomorphic mapping from the upper half-plane to D_1. Meanwhile, there exists a biholomorphic mapping from the unit disk to the upper half-plane. So we can obtain a biholomorphic mapping from the unit disk to a polygon. There are numerical methods[4] to compute this mapping and a Matlab toolbox[5] has been developed to numerically generate this mapping denoted by f^{-1} with its inverse f. Then we can directly use (1) to get the flow.

3.2.1 D is an inverse unit square

Suppose that $D = \{z \in \mathbb{C} : \max(|\mathrm{Re}(z)|, |\mathrm{Im}(z)|) < 1\}^{-1}$. The shape of D is like Figure 1.

Implement the method as below to get Figure 2:

[3] https://en.wikipedia.org/wiki/Schwarz%E2%80%93Christoffel_mapping
[4] http://i.stanford.edu/pub/cstr/reports/cs/tr/79/710/CS-TR-79-710.pdf
[5] https://github.com/tobydriscoll/sc-toolbox

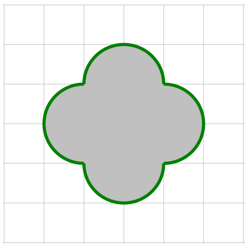


Figure 1: An inverse square.

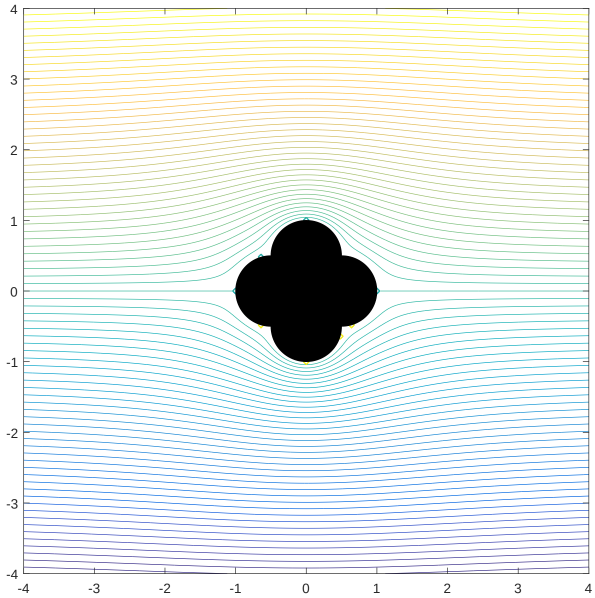


Figure 2: Uniform flow around an inverse square.

```
p=polygon([1+1i,-1+1i,-1-1i,1-1i]);%unit square
f=diskmap(p);%generate SC mapping
x=-4:0.04:4; y=-4:0.04:4;
G=zeros(length(y),length(x));
for m=1:length(x)
    for n=1:length(y)
        if max(abs(real(1/(x(m)+y(n)*1i))),abs(imag(1/(x(m)+y(n)*1i))))>1
            continue %skip calculations of points inside D
        end
        fz=evalinv(f,1/(x(m)+y(n)*1i));%calculate f(1/z)
        nfz=abs(fz); afz=angle(fz);
        g=@(phi)imag(1./f(exp(1i*phi)))./(1-2*nfz*cos(afz-phi)+nfz^2);
        G(n,m)=y(n)-0.5*(1-nfz^2)/pi*integral(g,0,2*pi);
    end
end
contour(x,y,G,'LevelStep',0.1);
rectangle('Position',[-1,-0.5,1,1],'Curvature',[1 1],'FaceColor',[0,0,0]);%plot the
    inverse square
rectangle('Position',[-0.5,0,1,1],'Curvature',[1 1],'FaceColor',[0,0,0]);
rectangle('Position',[-0.5,-1,1,1],'Curvature',[1 1],'FaceColor',[0,0,0]);
rectangle('Position',[0,-0.5,1,1],'Curvature',[1 1],'FaceColor',[0,0,0]);
axis equal;
```

Now change the direction of flow. As the uniform flow is set to $v_0 = y$, the direction of flow is fixed. But we can rotate the obstacle. Let $D = \{z \in \mathbb{C} : \max(|\mathrm{Re}(ze^{i\theta})|, |\mathrm{Im}(ze^{i\theta})|) < 1\}^{-1}$.

Take $\theta = \pi/4$ and $\theta = \pi/6$ in the following script to get Figure 3.

```
p=polygon([1+1i,-1+1i,-1-1i,1-1i]);%unit square
f=diskmap(p);%generate SC mapping
theta=pi/4;
x=-4:0.04:4; y=-4:0.04:4;
G=zeros(length(y),length(x));
for m=1:length(x)
    for n=1:length(y)
        z=(x(m)+y(n)*1i)*exp(1i*theta);
        if max(abs(real(1/z)),abs(imag(1/z)))>1
            continue %skip calculations of points inside D
        end
        fz=evalinv(f,1/z);%calculate f(1/z)
        nfz=abs(fz); afz=angle(fz);
        g=@(phi)imag(1./f(exp(1i*phi))*exp(-1i*theta))./(1-2*nfz*cos(afz-phi)+nfz^2);
        G(n,m)=y(n)-0.5*(1-nfz^2)/pi*integral(g,0,2*pi);
    end
end
contour(x,y,G,'LevelStep',0.1);
x1=real(0.5*exp(-1i*theta)); y1=imag(0.5*exp(-1i*theta));
x2=-y1;y2=x1;x3=-x1;y3=-y1;x4=-x2;y4=-y2;
rectangle('Position',[x1-0.5,y1-0.5,1,1],'Curvature',[1 1],'FaceColor',[0,0,0]);%plot
    the inverse square
rectangle('Position',[x2-0.5,y2-0.5,1,1],'Curvature',[1 1],'FaceColor',[0,0,0]);
rectangle('Position',[x3-0.5,y3-0.5,1,1],'Curvature',[1 1],'FaceColor',[0,0,0]);
rectangle('Position',[x4-0.5,y4-0.5,1,1],'Curvature',[1 1],'FaceColor',[0,0,0]);
axis equal;
```

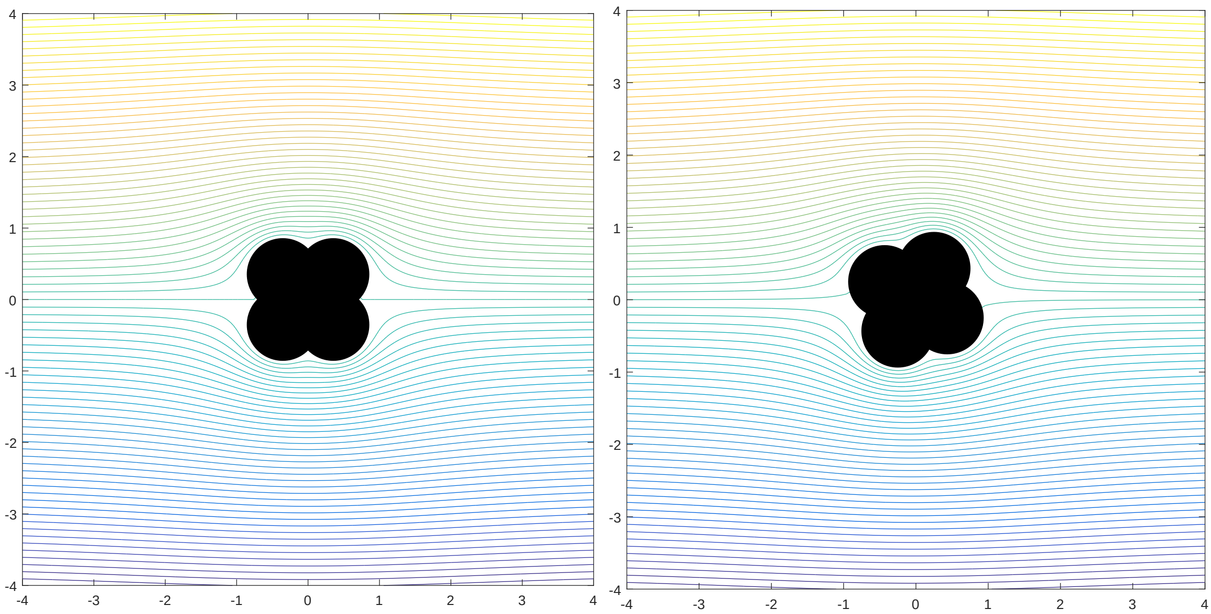


Figure 3: Uniform flow around an inverse square.

3.2.2 D is an inverse equilateral triangle

The equilateral triangle D_1 is defined as $D_1 = \{x+iy : (x,y) \in \mathbb{R}^2,\ y > -1 \text{ and } y < -\sqrt{3}|x|+2\}$. Take $\theta = 0, \pi/6$ in the following script to get Figure 4.

```
p=polygon([2i,-sqrt(3)-1i,sqrt(3)-1i]);%equilateral triangle
f=diskmap(p);%generate SC mapping
theta=pi/6;
x=-3:0.03:3; y=-3:0.03:3;
G=zeros(length(y),length(x));
for m=1:length(x)
    for n=1:length(y)
        z=(x(m)+y(n)*1i)*exp(1i*theta);
        if imag(1/z)<-1||imag(1/z)>2-sqrt(3)*abs(real(1/z))
            continue %skip calculations of points inside D
        end
        fz=evalinv(f,1/z);%calculate f(1/z)
        nfz=abs(fz);
        afz=angle(fz);
        g=@(phi)imag(1./f(exp(1i*phi))*exp(-1i*theta))./(1-2*nfz*cos(afz-phi)+nfz^2);
        G(n,m)=y(n)-0.5*(1-nfz^2)/pi*integral(g,0,2*pi);
    end
end
contour(x,y,G,'LevelStep',0.1);
x1=real(0.5i*exp(-1i*theta)); y1=imag(0.5i*exp(-1i*theta));
x2=real(0.5i*exp(-1i*(theta+2*pi/3))); y2=imag(0.5i*exp(-1i*(theta+2*pi/3)));
x3=real(0.5i*exp(-1i*(theta-2*pi/3))); y3=imag(0.5i*exp(-1i*(theta-2*pi/3)));
rectangle('Position',[x1-0.5,y1-0.5,1,1],'Curvature',[1 1],'FaceColor',[0,0,0]);
rectangle('Position',[x2-0.5,y2-0.5,1,1],'Curvature',[1 1],'FaceColor',[0,0,0]);
rectangle('Position',[x3-0.5,y3-0.5,1,1],'Curvature',[1 1],'FaceColor',[0,0,0]);
axis equal;
```

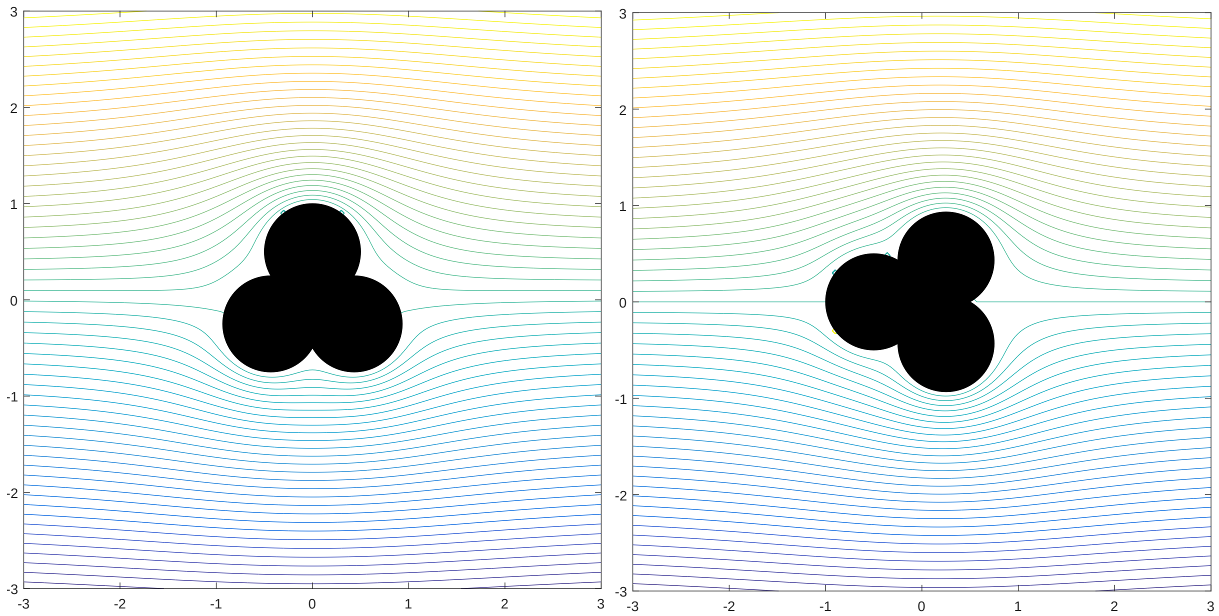


Figure 4: Uniform flow around an inverse equilateral triangle.

3.3 D is a simple polygon

As long as D is a polygon, D_1 is not a polygon and there is no direct method to obtain a mapping from D_1 to the unit disk. However, with Schwartz-Christoffel formula, we can directly obtain a biholomorphic mapping denoted by g from the exterior of D to the unit disk. Substitute $g(z) = f(1/z)$ into (1) and we get another formula:

$$\mathrm{Im}(\phi(z)) = y - \frac{1-|g(z)|^2}{2\pi}\int_0^{2\pi} \frac{\mathrm{Im}(g^{-1}(e^{i\varphi}))}{1-2|g(z)|\cos(\arg(g(z))-\varphi)+|g(z)|^2}\mathrm{d}\varphi. \tag{2}$$

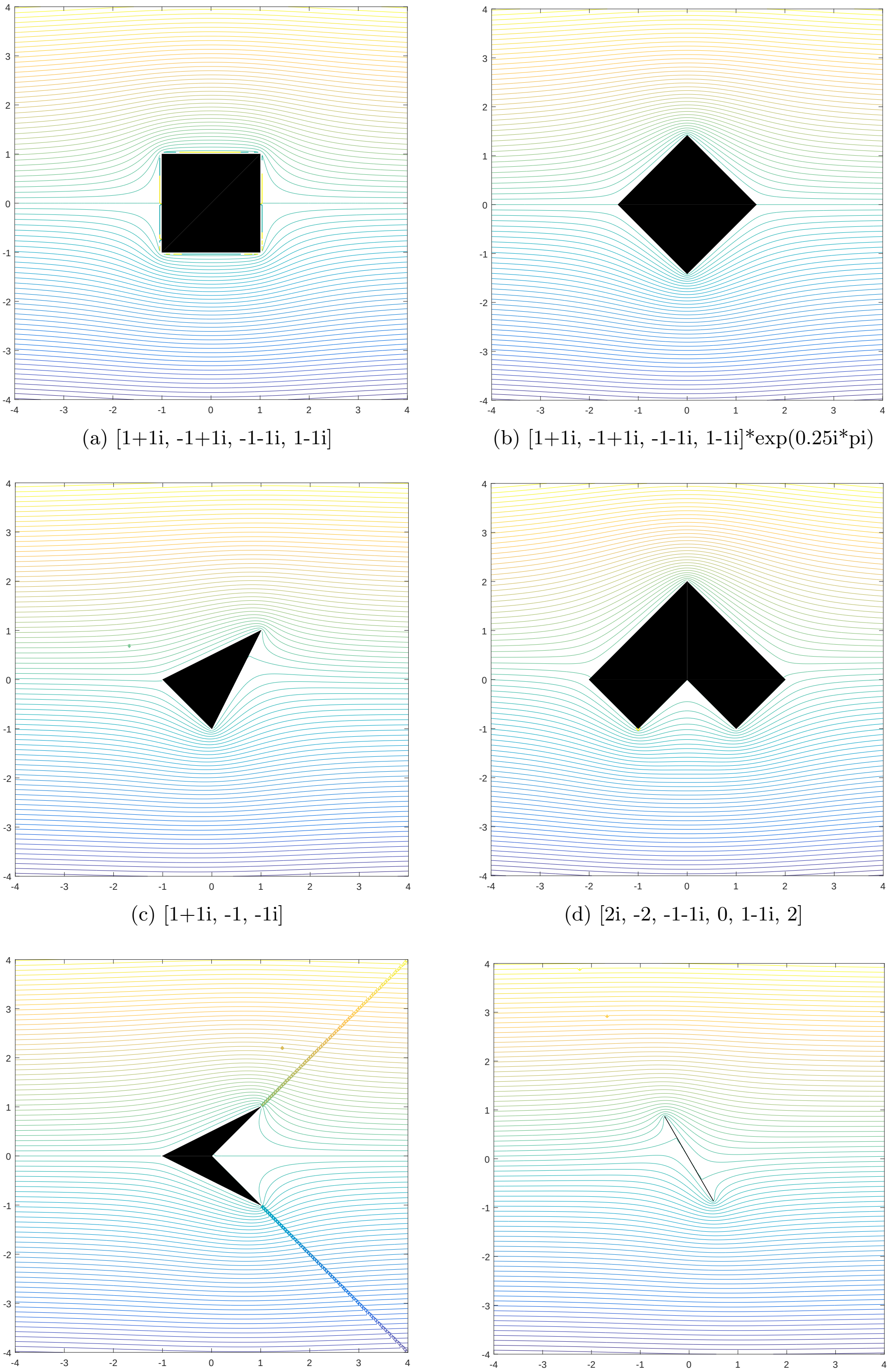
In this case, I made a function that can solve whatever simple polygon D.

```
function [ p,f,G ] = solveFluidPolygon(vertices,x,y)
% vertices: the complex vertices of polygon given in counterclockwise order
% x: grid on x-axis
% y: grid on y-axis
% p: polygon object
% f: SC mapping
% G: numerical solution on grids
p=polygon(vertices);
f=extermap(p);
G=zeros(length(y),length(x));
for m=1:length(x)
    for n=1:length(y)
        z=x(m)+y(n)*1i;
        if isinpoly(z,p)
            continue
        end
        fz=evalinv(f,z);
        nfz=abs(fz);
        afz=angle(fz);
        g=@(phi)imag(f(exp(1i*phi)))./(1-2*nfz*cos(afz-phi)+nfz^2);
        G(n,m)=y(n)-0.5*(1-nfz^2)/pi*integral(g,0,2*pi);
    end
end
contour(x,y,G,'LevelStep',0.1);
hold on
fill(real(vertices),imag(vertices),[0,0,0]);
axis equal;
end
```

Figure 5 are examples on how to use this function. Figure 5(f) shows that although this method is designed for open D, it can be applied to a much wider range of situations.

4 General cases

In cases when D is just a general simply connected open subset of $\mathbb{C}$, finding Riemann mapping is difficult. However, as the problem is well-posed in terms of ∂D, we can use a polygon to estimate the boundary of D. Let us try this method on a unit disk as the case in subsection 3.1. The results in Figure 6 get closed to that of subsection 3.1 as edges of the polygon increases.



(a) [1+1i, -1+1i, -1-1i, 1-1i]

(b) [1+1i, -1+1i, -1-1i, 1-1i]*exp(0.25i*pi)

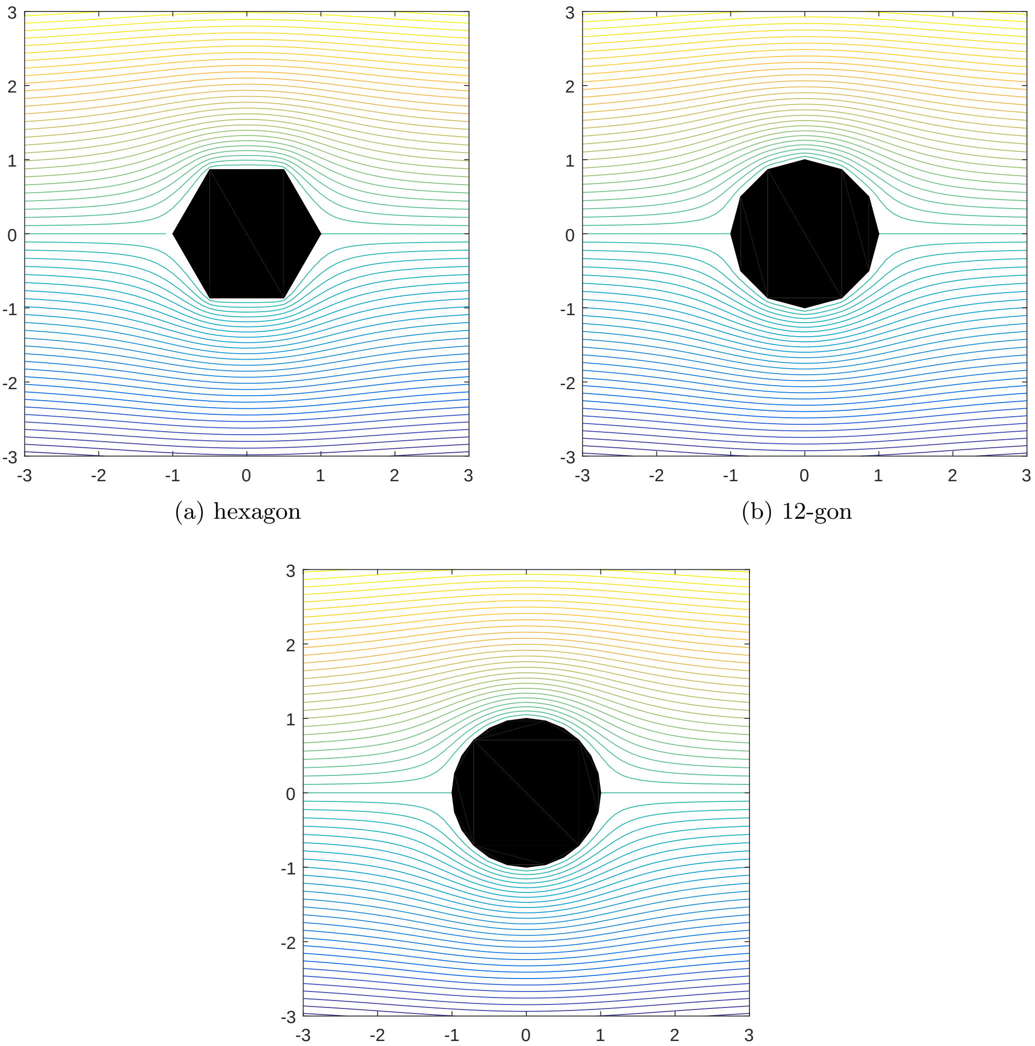
(c) [1+1i, -1, -1i]

(d) [2i, -2, -1-1i, 0, 1-1i, 2]

(e) [1+1i,-1,1-1i,0]

(f) [1i,0,-1i,0]*exp(1i*pi/6)

Figure 5: Ideal uniform fluid around various of polygons. Both of the arguments x and y are -4:0.04:4 .



(a) hexagon

(b) 12-gon

(c) 24-gon

Figure 6

5 Ideal uniform fluid flow around multiple obstacles

The method developed here not only has such a high complexity, but it may not converge as well. To see how this method works, let us first consider a simple case.

5.1 Ideal uniform fluid around two disks

Here we choose $D = B(1,1) \cup B(-1,1)$. To find a complex function ψ whose imaginary part cancels $\mathrm{Im}(z)$ on ∂D, we find ψ_{l1} whose imaginary part cancels $\mathrm{Im}(z)$ on the left circle and ψ_{r1} whose imaginary part cancels $\mathrm{Im}(z)$ on the right circle. Then we have the approximation $\psi \approx \psi_{l1} + \psi_{r1}$. This is not the accurate solution since ψ_{l1} creates additional terms on the right circle and ψ_{r1} creates additional terms on the left circle.

To cancel these additional terms, we introduce ψ_{l2} whose imaginary part cancels that of ψ_{r1} on the left circle and ψ_{r2} whose imaginary part cancels that of ψ_{l1} on the right circle. Repeat the steps and we get a series $\psi = \sum_{n=1}^{\infty}(\psi_{ln} + \psi_{rn})$. To find the exact expressions of ψ_{ln}'s and

ψ_{rn}'s, we introduce the following proposition, the proof of which is trivial.

Proposition 1 *(i)The imaginary parts of z and $1/z$ cancel each other on the unit circle. (ii)If $z_0 \in \mathbb{R}$, then the imaginary parts of $\frac{1}{z-z_0}$ and $-\frac{1}{z_0^2(z-1/z_0)}$ cancel each other on the unit circle.*

By proposition 1(i),

$$\psi_{l1} = \frac{1}{z+1} \quad \text{and} \quad \psi_{r1} = \frac{1}{z-1}.$$

Suppose that

$$\psi_{ln} = \frac{(-1)^{n-1}}{n^2(z+1/n)} \quad \text{and} \quad \psi_{rn} = \frac{(-1)^{n-1}}{n^2(z-1/n)}.$$

For z on the left circle, $z+1$ is on the unit circle. Since $\psi_{rn} = \frac{(-1)^{n-1}}{n^2(z-1/n)} = \frac{(-1)^{n-1}}{n^2(z+1-(n+1)/n)}$, by proposition 1(ii),

$$\psi_{l,n+1} = \frac{(-1)^n}{n^2}\frac{n^2}{(n+1^2)}\frac{1}{z+1-n/(n+1)} = \frac{(-1)^n}{(n+1)^2(z-1/(n+1))}.$$

Similarly,

$$\psi_{r,n+1} = \frac{(-1)^n}{(n+1)^2(z-1/(n+1))}.$$

By induction,

$$\psi_{ln} = \frac{(-1)^{n-1}}{n^2(z+1/n)} \quad \text{and} \quad \psi_{rn} = \frac{(-1)^{n-1}}{n^2(z-1/n)}.$$

Thus

$$\psi = \sum_{n=1}^{\infty}(\psi_{ln}+\psi_{rn}) = \sum_{n=1}^{\infty}\frac{(-1)^{n-1}}{n^2}(\frac{1}{z+1/n}+\frac{1}{z-1/n}).$$

This series converges everywhere on the exterior of D. Take partial sum of the first N terms as approximation and we get Figure 7.

5.2 General cases

For two general obstacles, the idea is similar as for two disks. Let D_1 and D_2 denote the obstacles. Assume that we can obtain the solutions of Laplace equations in the form

$$\Delta v = 0, \quad (x,y) \in \mathbb{R}^2 \backslash D_j$$

$$v(x,y) = f(x,y), \quad (x,y) \in \partial D_j$$

$$v(\infty,\infty) \quad \text{exists.}$$

where $j=1,2$ and f is any bounded function on ∂D_j. Let $v[f,j]$ denote the solution associated with f and D_j. Define $v_j^1 = -v[y,j]$ and $v_j^n = -v[v_{3-j}^{n-1},j]$ for $j=1,2$, $n=2,3,4,\cdots$. Let $v = y + \sum_{n=1}^{\infty}(v_1^n + v_2^n)$, then v is the solution whenever RHS exists. However, whether the series on RHS converges remains unknown for general problems.

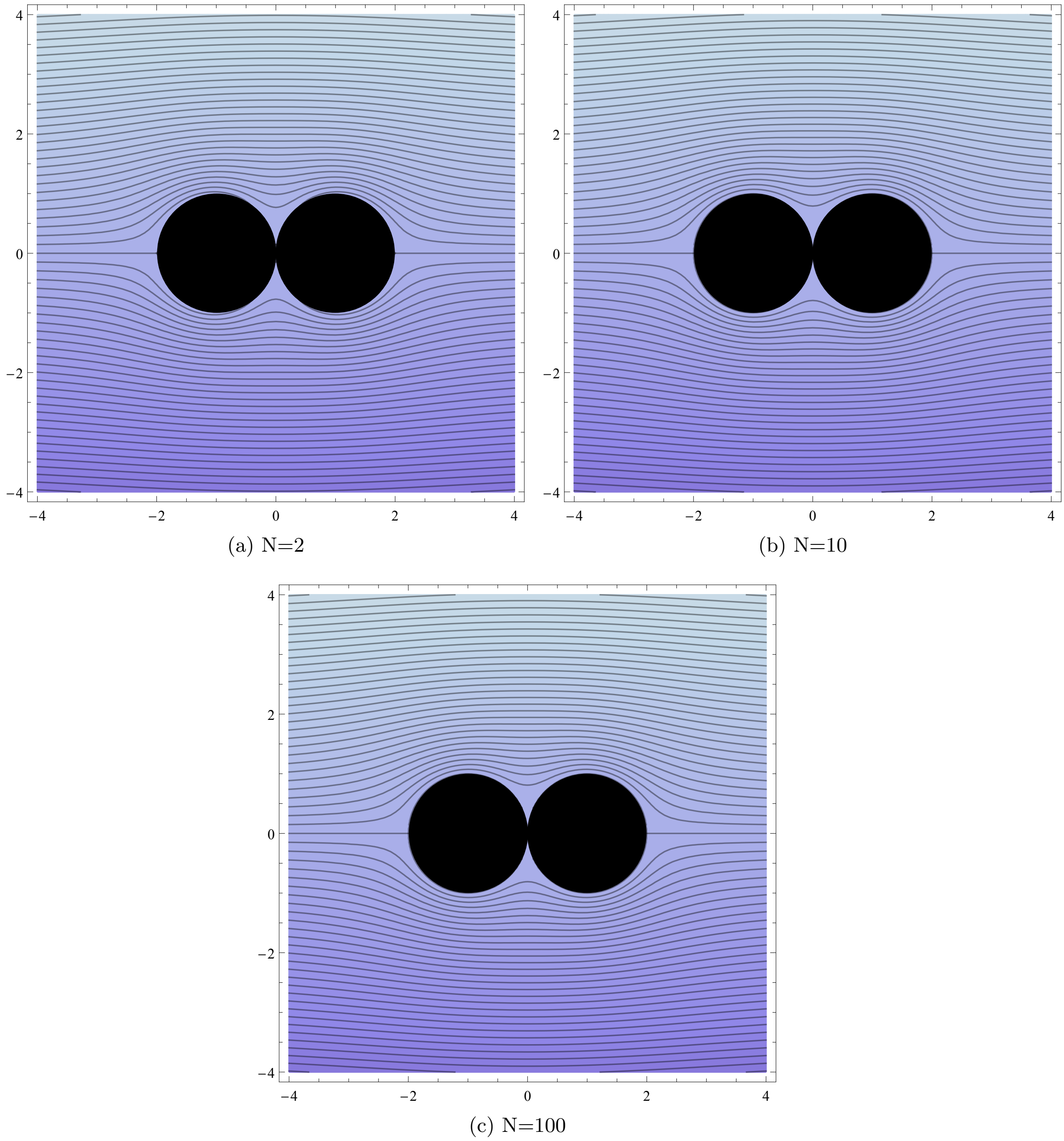


Figure 7: Approximation of ideal uniform fluid flow around D.

6 Conclusions

In this project, I developed methods to solve a 2D ideal uniform fluid flow around a single obstacle of some specific shapes. I also came up with a convergent method to solve the flow around a single obstacle of general shape and shared an idea to solve the flow around multiple obstacles.

It is straightforward to apply method in LectureNote#12 to solve the problems. However, although we can generate a mapping from the exterior of D to the upper half plane, we cannot use Cauchy transform or Hilbert transform because the solution from them does not decay on the original domain. In other words, they can be used to generate ψ whose imaginary part cancels that of z, but they do not guarantee that $\phi(z) \sim z$ as $z \to \infty$. Moreover, the method in LectureNote#12 performs two integrals during the inverse Hilbert transform and Cauchy transform, compared to methods in this project which performs only one integral. One advantage of the method in LectureNote#12 is that it can solve problems for multiple line segments, while the methods in this project cannot.