Numerical approximation of potential flow around a plate by potential flows around an ellipse

Franklin Pezzuti Dyer, with faculty mentor Monika Nitsche

December 2021

Abstract

In this report, we briefly describe the theory of potential flows from fluid dynamics, and introduce point vortices as a tool for constructing more complex flows with desired properties. We then derive formulae that allow us the approximate potential flow past an elliptical cylinder, and use the limiting behavior of this flow as the ellipse becomes thinner to approximate potential flow past a plate. We also perform a basic error analysis of this approximation method, and notice how cusplike behavior gives rise to very slow convergence at certain points near the plate.

1 Background

In fluid dynamics, **potential flow** is a special type of fluid flow defined by certain assumptions placed on a fluid's behavior. In particular, potential flow is assumed to be **ideal** or **inviscid**, as well as **irrotational**. Intuitively, inviscid flow can be understood as fluid flow in which the viscous forces resisting deformation are negligible compared to inertial forces which "push" a fluid along the same trajectory that it is currently following. Irrotational flow has zero vorticity or "local spinning motion" anywhere in its domain. Consequently, potential flow has the additional property that it doesn't ever "flow in circles." (Stated more precisely, on a simply connected domain, the circulation of a potential flow around any closed curve is 0, where **circulation** is a measure of the extent to which a flow moves in the "same direction" as another curve.) [1]

There is a special type of potential flow induced by something known as a **point vortex**, which can be thought of as a force that induces fluid to flow in circles like a "whirlpool". The point vortex with strength Γ is defined by the velocity field

$$\mathbf{v}(x, y) = \frac{\Gamma}{2\pi} \frac{\langle -y, x \rangle}{x^2 + y^2}$$

This velocity field induces fluid flow in counterclockwise circles around the origin $(0, 0)$. This may seem to contradict the irrotational property of potential

flow - however, it turns out that the above velocity field is undefined at the origin $(x, y) = (0, 0)$ since $\mathbf{v}$ does not have a limit as (x, y) approaches $(0, 0)$. This means that the domain of definition of this potential flow is not the entire plan $\mathbb{R}^2$, but rather the "punctured plane" $\mathbb{R}^2 \backslash (0, 0)$, or the plane minus the point at $(0, 0)$. This is *not* a simply connected domain. The circulation about any simple closed curve surrounding the origin is equal to Γ - but if a curve does not surround the origin, it exists in a simply connected part of the fluid's domain, meaning that it has zero circulation.

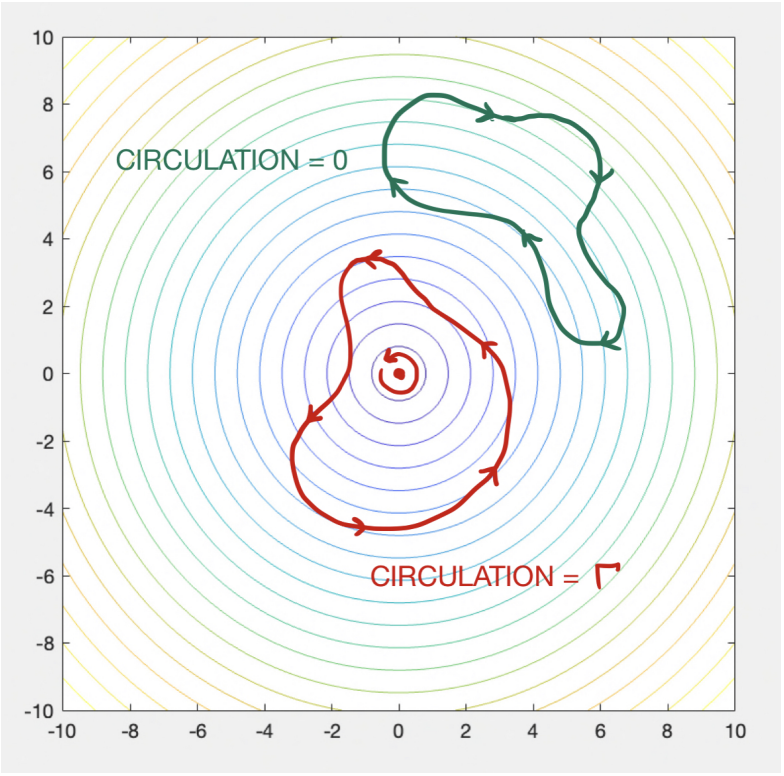


Figure 1: Circulation around various paths surrounding a simple point vortex.

We can also consider fluid flows induced by two or more point vortices. The velocity fields of these flows can be calculated by simply adding together the velocity fields induced by each of the point vortices. For instance, a flow induced by two point vortices gyrating in the same direction might look like this:

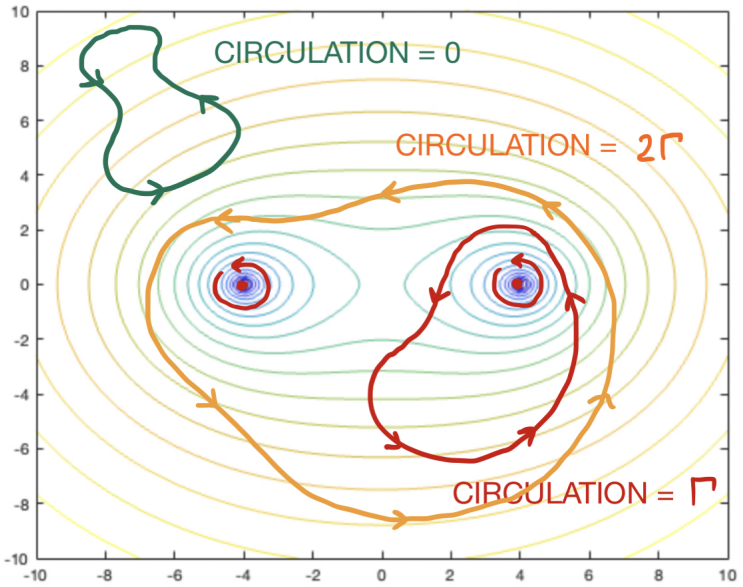


Figure 2: Circulation around various paths surrounding two point vortices of the same orientation.

whereas the flow induced by two point vortices gyrating in opposite directions might look like this:

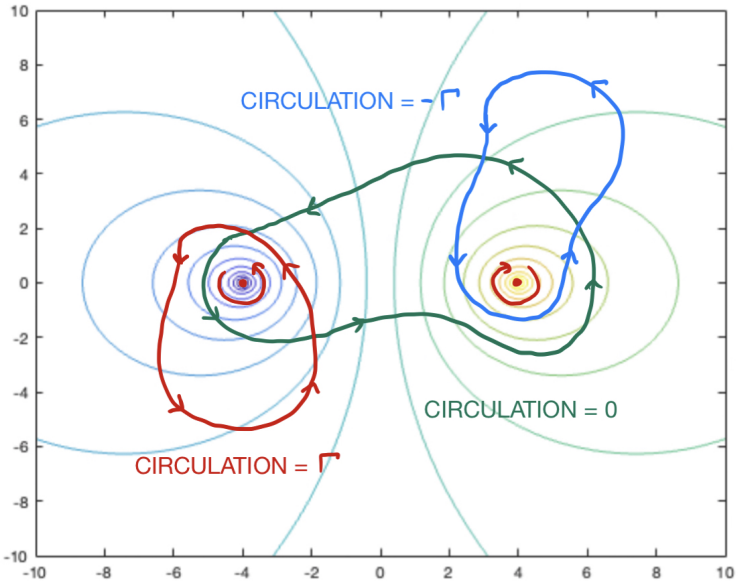


Figure 3: Circulation around various paths surrounding two point vortices of different orientations.

For more examples of potential flows induced by various combinations of point vortices, see Appendix A (in which this type of point vortex is referred to as a "v-slow" vortex).

Using techniques from calculus, we are able to transition from discrete sets of point vortices, each having some finite vortex strength, to continua consisting of infinitely many point vortices with infinitesimal strengths distributed across them. In this way, we can construct flows in which vorticity is generated by some continuous shape, rather than single isolated points. For instance, consider the following flow consisting of 10 point vortices of equal strength arranged in a straight line:

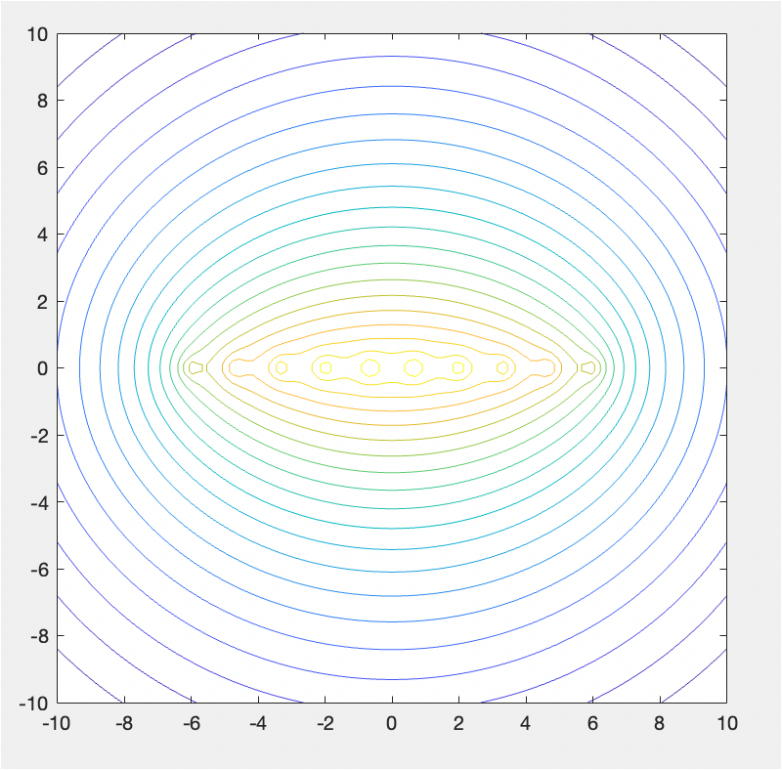


Figure 4: Potential flow around ten point vortices arranged in a straight line.

Because there are so many vortices so tightly packed together, the small ringlike trajectories surrounding each vortex are hardly visible. If we increase the number of point vortices to 20, they are so tightly packed together that the flow seems to be shaped by continous line segment of stagnation, rather than a collection of many discrete point vortices:

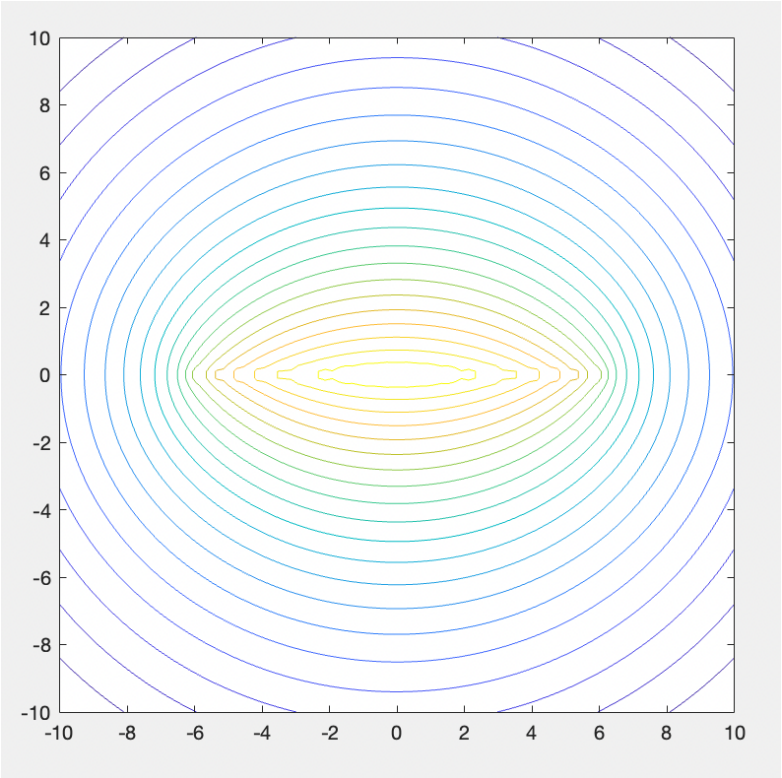


Figure 5: Potential flow around twenty point vortices arranged in a straight line.

By strategically choosing the positions and strengths of point vortices, we can enforce the existence of certain regions of stagnation in the resulting flow. This is how, for instance, we will later approximate potential flow around a circular cylinder: by placing a large number of point vortices in a circular arrangement with particular vortex strengths, we cause a cancellation in the induced velocity field at points on the circumference of the circle, so that flow never "pushes" or "pulls" perpendicular to the circumference, only "sliding" parallel to it. This is meant to model how fluid flows past solid objects, so that this configuration models how an ideal, irrotational fluid would flow past a solid cylindrical object. (Other models of fluid flow give rise to other desirable behaviors, such as **Stokes flow**, in which fluid flow is zero on boundaries - so that, unlike in potential flow, fluid is not allowed to "slip" past surfaces either.)

To describe this construction more precisely: the velocity field induced by a point vortex at the point (x_0, y_0) is given by

$$\mathbf{v}(x,y) = \frac{\Gamma}{2\pi} \frac{\langle -(y-y_0), (x-x_0) \rangle}{(x-x_0)^2 + (y-y_0)^2}$$

[2] From this, it follows that if we have a sequence of finitely many point

vortices with strengths $\Gamma_1, ..., \Gamma_n$ situated at the points $(x_1, y_1), ..., (x_n, y_n)$ respectively, the overall velocity field is given by

$$\mathbf{v}(x, y) = \sum_{k=1}^{n} \frac{\Gamma_i}{2\pi} \frac{\langle -(y - y_k), (x - x_k) \rangle}{(x - x_k)^2 + (y - y_k)^2}$$

Now, if we situate these vortices along the length of some curve C evenly spaced along its length L, so that the relative strength of each vortex Γ is a function of its position $\Gamma(x_k, y_k)$ and inversely proportional to the number of vortices (so that their aggregrate strength does not grow unboundedly large), we have that

$$\mathbf{v}(x, y) = \sum_{k=1}^{n} \frac{\Gamma(x_k, y_k)}{2\pi n} \frac{\langle -(y - y_k), (x - x_k) \rangle}{(x - x_k)^2 + (y - y_k)^2}$$

This takes the form of the Riemann Sum, and $\Gamma(x_k, y_k)/n$ measures the "point vortex strength density", which we may also denote as $d\Gamma$, at a point along the curve. As the number of vortices n becomes increasingly large and grows towards infinity, the above Riemann Sum approaches the integral

$$\mathbf{v}(\mathbf{x_0}) = \frac{1}{2\pi} \int_C \frac{\mathbf{k} \times (\mathbf{x}_0 - \mathbf{x})}{||\mathbf{x}_0 - \mathbf{x}||^2} d\Gamma$$

Sometimes we wish to consider how an object behaves when placed in a preexisting potential flow or a **background flow u**, we add this term to the above expression, so that we have the formula

$$\mathbf{v}(\mathbf{x_0}) = \frac{1}{2\pi} \int_C \frac{\mathbf{k} \times (\mathbf{x}_0 - \mathbf{x})}{||\mathbf{x}_0 - \mathbf{x}||^2} d\Gamma + \mathbf{u}(\mathbf{x}_0)$$

In this report, we consider steady flow past a cylinder in which the background flow is moving at a constant rate in the vertical direction, so that $\mathbf{u} = \langle 0, u_\infty \rangle$ everywhere. In the following section, we will "reverse-engineer" the above formula to solve for some distribution Γ of point vortex strengths that results in a potential flow whose velocity component normal to a circular cylinder equals 0.

In some cases, we might want to consider potential flow past an object which has corners or cusps and is not smooth like a circle, such as a polygon or a line segment. However, these types of surfaces do not necessarily have well-defined normal vectors at all points, since may be impossible to parametrize with differentiable functions. However, we can consider potential flows path smooth surfaces which are "near" in shape to the non-smooth surface in question, and use them as an approximation of the flow past that surface.

The purpose of this report is to explore a particular simple case of this technique. Namely, we consider potential flow past sequences of ellipses with one semiradius shrinking towards 0, so that they become increasingly thin, becoming closer and closer in shape to a flat line segment. In this way, we estimate

potential flow past a flat plate by approximating the plate by a sequence of thin ellipses.

In the following sections, we derive the specific integral equations used to solve for Γ in the case of potential flow past an ellipse, and solve these equations approximately in MATLAB. Then we test our code on a few simple examples and evaluate its performance as the ellipses approach the shape of a plate.

2 Methods

As discussed in the previous section, at any point on the boundary of a smooth surface, the normal component of the potential flow must equal zero, which follows from incompressibility. In the constant background flow $\langle 0, u_\infty \rangle$ past a smooth curve C with point vortex strength densities given by Γ, the velocity at a point $\mathbf{x}_0$ can be written as

$$\frac{1}{2\pi} \int_C \frac{\mathbf{k} \times (\mathbf{x}_0 - \mathbf{x})}{||\mathbf{x}_0 - \mathbf{x}||^2} d\Gamma + \langle 0, u_\infty \rangle$$

We may therefore formulate the zero-normal-velocity condition in integral form as follows: if $\mathbf{x}_0$ is a point on the curve C, and $\mathbf{n}_0$ is the normal vector to C at that point, then

$$\frac{1}{2\pi} \int_C \frac{\mathbf{n}_0 \cdot \mathbf{k} \times (\mathbf{x}_0 - \mathbf{x})}{||\mathbf{x}_0 - \mathbf{x}||^2} d\Gamma = \mathbf{n}_0 \cdot \langle 0, -u_\infty \rangle \quad \forall \mathbf{x}_0 \in C$$

Notice that this is a **singular integral**: because $\mathbf{x_0}$ is on C and the variable of integration $\mathbf{x}$ ranges over C, we have that $\mathbf{x}_0 - \mathbf{x} = \mathbf{0}$ and the denominator of the integrand becomes 0 for some $\mathbf{x} \in C$. Despite this singularity, the integral can still be calculated validly using the **Cauchy principal value**. (The problem of numerically dealing with singular or near-singular integrals often comes up when computing fluid flows. In fact, my faculty mentor has devised a novel method for evaluating near-singular integrals that arose in her computations of vortex sheet flows past a plate. [3]) We will soon discuss how this problem is addressed numerically in our code.

In the case where C is an ellipse with semiradii 1 and s, we may parametrize it as follows:

$$\mathbf{x} = \langle \cos\theta, s \sin\theta \rangle$$

so that the above identity becomes

$$\begin{aligned}
\frac{1}{2\pi}\int_\gamma \frac{\mathbf{n}_0 \cdot \mathbf{k} \times (\mathbf{x}_0 - \mathbf{x})}{||\mathbf{x}_0 - \mathbf{x}||^2} d\Gamma &= \frac{1}{2\pi}\int_0^{2\pi} \frac{\mathbf{n}_0 \cdot \mathbf{k} \times (\langle x_0, y_0\rangle - \langle \cos\theta, s\sin\theta\rangle)}{||\langle x_0, y_0\rangle - \langle \cos\theta, s\sin\theta\rangle||^2} d\Gamma \\
&= \frac{1}{2\pi}\int_0^{2\pi} \frac{\mathbf{n}_0 \cdot \mathbf{k} \times (\langle x_0 - \cos\theta, y_0 - s\sin\theta\rangle)}{||\langle x_0 - \cos\theta, y_0 - s\sin\theta\rangle||^2} d\Gamma \\
&= \frac{1}{2\pi}\int_0^{2\pi} \frac{\mathbf{n}_0 \cdot (\langle y_0 - s\sin\theta, \cos\theta - x_0\rangle)}{(x_0 - \cos\theta)^2 + (y_0 - s\sin\theta)^2} d\Gamma \\
&= \mathbf{n}_0 \cdot \langle 0, -u_\infty\rangle
\end{aligned}$$

For the ellipse in question, if $\Gamma(\theta)$ is the point vortex strength density as a function of θ, and the infinitesimal point vortices comprising the ellipse are distributed uniformly by length, we have that

$$\mathbf{t} = \langle -\sin\theta, s\cos\theta\rangle$$

$$\mathbf{n}_0 = \langle s\cos\theta_0, \sin\theta_0\rangle$$

$$d\Gamma = \frac{\Gamma(\theta)d\theta}{||\mathbf{t}||} = \frac{\Gamma(\theta)d\theta}{\sqrt{\sin^2\theta + s^2\cos^2\theta}}$$

Using these facts, we have that

$$\begin{aligned}
\frac{1}{2\pi}\int_\gamma \frac{\mathbf{n}_0 \cdot \mathbf{k} \times (\mathbf{x}_0 - \mathbf{x})}{||\mathbf{x}_0 - \mathbf{x}||^2} d\Gamma &= \int_0^{2\pi} \frac{\mathbf{n}_0 \cdot (\langle y_0 - s\sin\theta, \cos\theta - x_0\rangle)}{(x_0 - \cos\theta)^2 + (y_0 - s\sin\theta)^2} d\Gamma \\
&= \frac{1}{2\pi}\int_0^{2\pi} \frac{s\cos\theta_0(y_0 - s\sin\theta) + \sin\theta_0(\cos\theta - x_0)}{(x_0 - \cos\theta)^2 + (y_0 - s\sin\theta)^2} d\Gamma \\
&= \frac{1}{2\pi}\int_0^{2\pi} \frac{s\cos\theta_0(y_0 - s\sin\theta) + \sin\theta_0(\cos\theta - x_0)}{(x_0 - \cos\theta)^2 + (y_0 - s\sin\theta)^2} \frac{\Gamma(\theta)d\theta}{\sqrt{\sin^2\theta + s^2\cos^2\theta}} \\
&= -u_\infty \sin\theta_0
\end{aligned}$$

If we can solve for the vortex strength density function Γ, then we can use it to calculate the velocity field at any point using the above integrals. Thus, we will attempt to numerically solve the following system of integral equations for Γ:

$$\frac{1}{2\pi}\int_0^{2\pi} \frac{s\cos\theta_0(y_0 - s\sin\theta) + \sin\theta_0(\cos\theta - x_0)}{(x_0 - \cos\theta)^2 + (y_0 - s\sin\theta)^2} \frac{\Gamma(\theta)d\theta}{\sqrt{\sin^2\theta + s^2\cos^2\theta}} = -u_\infty \sin\theta_0$$

$$\frac{1}{2\pi}\int_0^{2\pi} \frac{\Gamma(\theta)d\theta}{\sqrt{\sin^2\theta + s^2\cos^2\theta}} = \gamma$$

where γ is a free parameter measuring the circulation of the flow around the boundary of the ellipse.

To accomplish this in MATLAB, we discretize both the interval of integration and the continuum of point vortices, using points equally spaced in θ around the boundary of the ellipse. In order to deal with the singular integral in the first of these equations, we use different sets of θ-values for the variable of integration and the point vortices. In particular, the variable of integration ranges over θ-values starting at $\theta = 0$ and increasing in increments of $2\pi/n$, and the point vortices are situated at points starting at $\theta = \pi/n$ and increasing in increments of $2\pi/n$. In MATLAB, their θ-values are calculated as follows:

```
% Calculation of angles
% "thetas" are the gridpoints of integration, starting at 0
% "theta0s" are the point vortex locations, shifted by dtheta/2
dtheta = 2*pi/n;
theta0s = linspace(0, 2*pi, n+1);
theta0s(end) = [];
thetas = theta0s;
theta0s = theta0s + dtheta/2;
```

For convenience, we also calculate the x- and y-values of the gridpoints of integration and the discrete point vortices, as well as the values of $||\mathbf{t}||$ involved in the integral:

```
% Calculation of coordinates
% "xs" and "ys" are gridpoints for integration
% "speeds" are the arclength per unit angle at each point
% "nxs" and "nys" are unit normal vecs at each point
% Analogous values for the theta0s are calculated
xs = cos(thetas);
ys = semirad*sin(thetas);
speeds = sqrt(semirad^2*cos(thetas).^2+sin(thetas).^2);
nxs = semirad*cos(thetas)./speeds;
nys = sin(thetas)./speeds;
speed0s = sqrt(semirad^2*cos(theta0s).^2+sin(theta0s).^2);
nx0s = semirad*cos(theta0s)./speed0s;
ny0s = sin(theta0s)./speed0s;
x0s = cos(theta0s);
y0s = semirad*sin(theta0s);
```

In MATLAB, we represent $\Gamma(\theta)$ as a vector $\mathbf{\Gamma}$ containing the point vortex strengths at each of the discretized vortices. The first integral equation is represented in the form

$$\mathbf{A\Gamma} = -\mathbf{u}$$

where

$$\mathbf{A}_{ij} = \frac{1}{2\pi n} \frac{s\cos\theta_{0i}(y_{0i} - s\sin\theta_j) + \sin\theta_{0i}(\cos\theta_j - x_{0i})}{(x_{0i} - \cos\theta_j)^2 + (y_{0i} - s\sin\theta_j)^2} \frac{1}{\sqrt{\sin^2\theta_j + s^2\cos^2\theta_j}}$$

$$\mathbf{u}_i = u_\infty \sin\theta_{0i}$$

in which θ_j denotes the jth gridpoint of integration, and θ_{0i}, x_{0i}, and y_{0i} represent the position of the ith point vortex. We add an extra row to both the matrix $\mathbf{A}$ and the vector $\mathbf{u}$ to include the second integral equation containing the parameter γ for circulation around the ellipse. In MATLAB, all of this is implemented as follows:

```
% Set up matrix system for integral equation
int_matrix = [];
for j=1:n
    theta0 = theta0s(j);
    x0 = x0s(j);
    y0 = y0s(j);
    denom = ((xs-x0).^2+(ys-y0).^2);
    numer = nx0s(j) .* (ys - y0) + ny0s(j) .* (-xs + x0);
    int_weight = (1/(2*pi*n))*numer./(denom.*speeds);
    int_matrix = [int_matrix; int_weight];
end

int_matrix = [int_matrix; ones(1, n)/n];
conds = [conds, cond(int_matrix)];
normal_vel = -u_infty*ny0s;

% Add an extra equation for the circulation around the ellipse
normal_vel = [normal_vel, circ];
```

Finally, we solve for Γ:

```
% Solve the system
Gamma = int_matrix\normal_vel';
```

In the following section, we will analyze the performance of this method as follows:

- First, we calculate Γ and the potential flow for a few simple cases, such as flow around a cylinder.
- Next, we analyze how these approximations of Γ and the velocity field behave as n increases.
- Finally, we observe the behavior of Γ and the convergence rate as the smaller semiradius $s \to 0$ and the ellipse approaches the shape of a thin plate.

3 Numerical results

We start by testing the above method for a few simple, well-known examples, starting with zero-circulation potential flow around a circular cylinder. Generating and plotting Γ using the above algorithm with $n = 200$ yields the graph shown below:

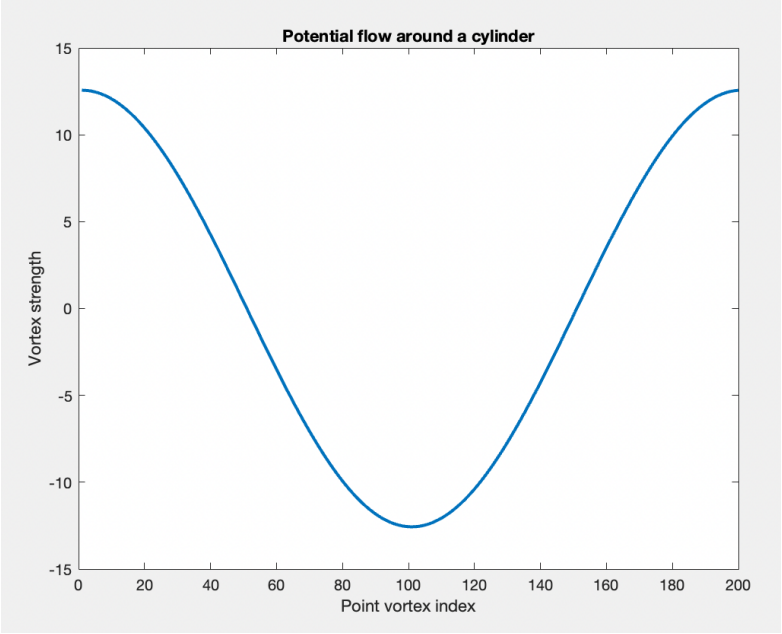


Figure 6: Point vortex strength distribution for potential flow around a circular cylinder.

Note that this matches with the analytical value of $\Gamma(\theta) = 4\pi \cos\theta$, corroborating the validity of our methods. When we generate a velocity field and streamlines from the calculated Γ, we obtain the following plot:

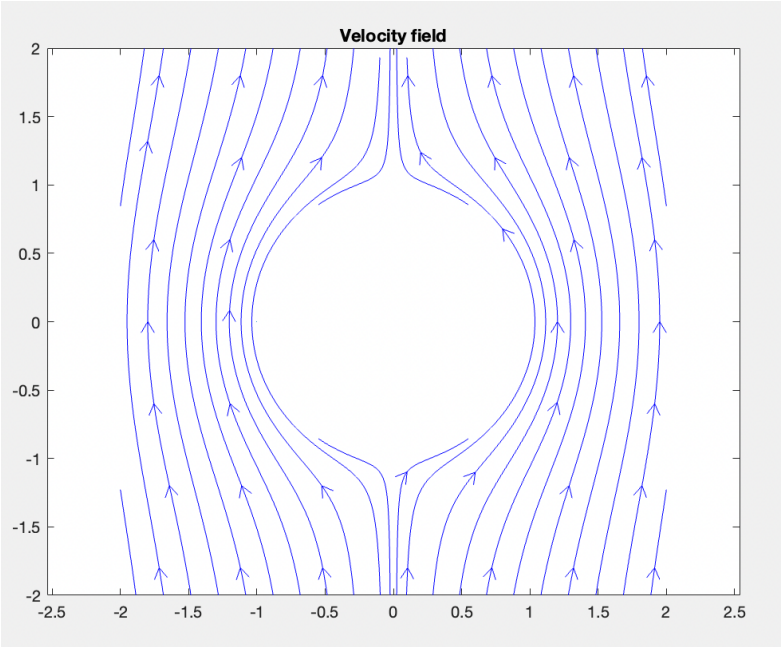


Figure 7: Streamlines for potential flow around a circular cylinder.

Running the same computations with a different circulation value of $\gamma = 3$ results in the following streamlines instead:

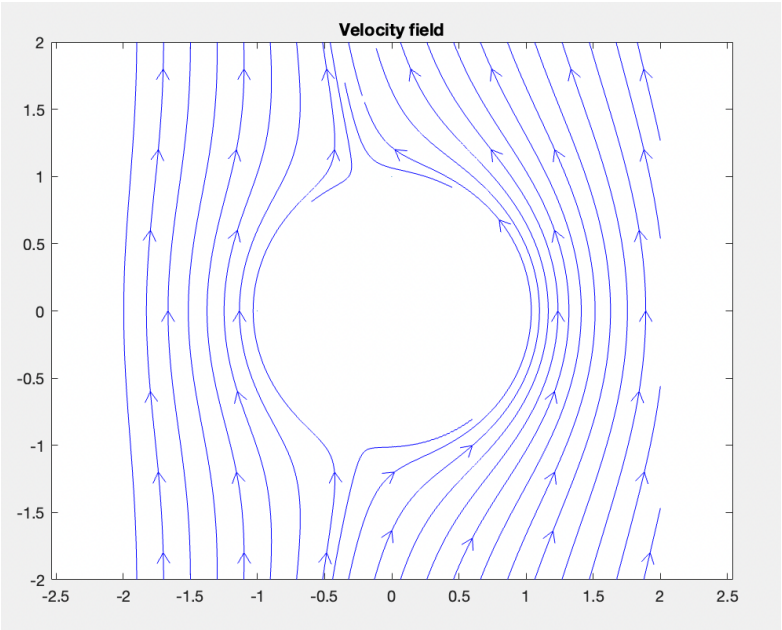


Figure 8: Streamlines for potential flow around a circular cylinder with nonzero circulation.

As we can see, the flow is now slightly "bent" so that its counterclockwise circulation about the circular cylinder is greater than its clockwise circulation, as we would expect. We can also obtain the streamlines of potential flow past an elliptical cylinder, say, with a semiradius of $s = 0.33$:

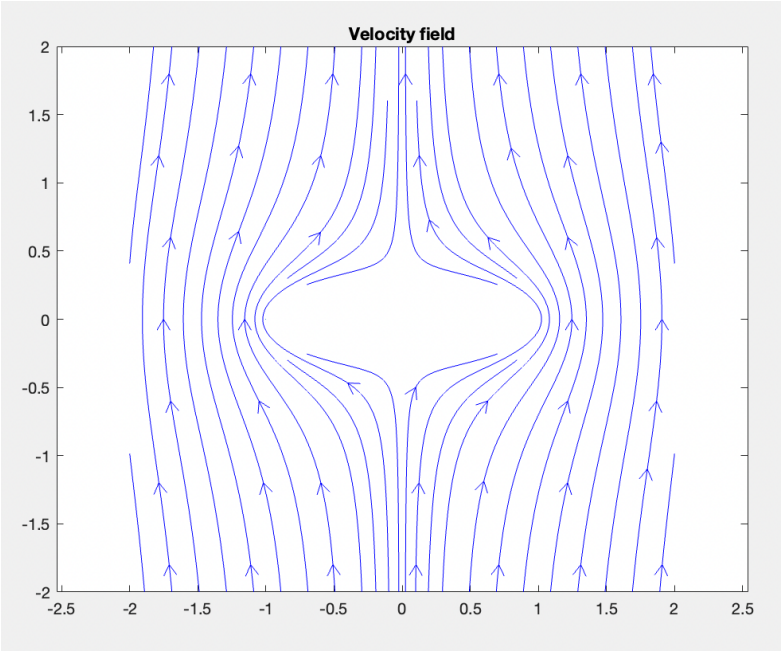


Figure 9: Streamlines for potential flow around an elliptical cylinder with semiradius 0.33.

and the plotted sequence of point vortex strengths $\mathbf{\Gamma}$ for the above flow appears as follows:

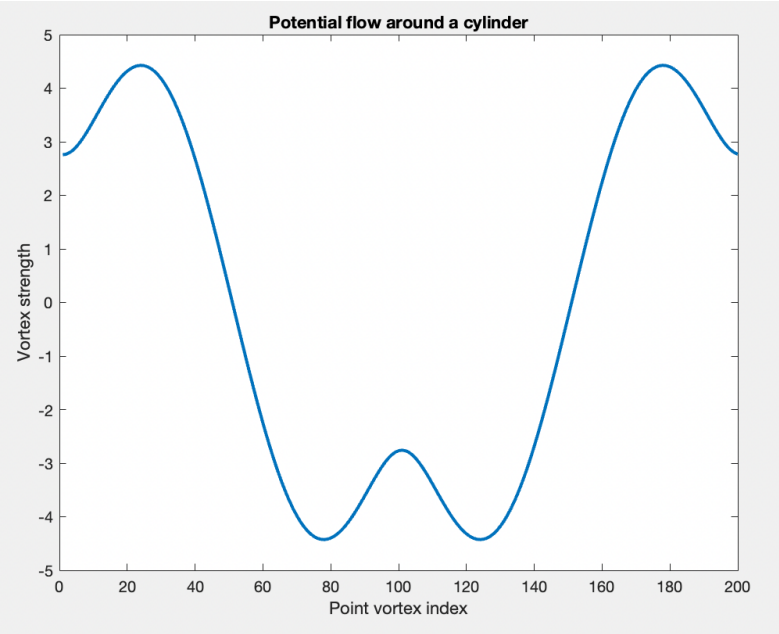


Figure 10: Point vortex strength distribution for potential flow around an elliptical cylinder with semiradius 0.33.

Note that the greatest oscillations in the point vortex strengths occur around $\theta = 0$ and $\theta = \pi$, at the points on the far left and right sides of the ellipse. Later, when we consider the limit as $s \to 0$ in which the ellipse approximates a flat plate, we will see how this phenomenon is amplified as the semiradius of the ellipse decreases.

Knowing the analytic solution $\Gamma(\theta) = 4\pi \cos\theta$ for potential flow around a circular cylinder with background flow $u_i nfty = 1$ and circulation $\gamma = 0$, we can also evaluate the error in our approximation of the velocity field. We will approximate the error by calculating

$$|\mathbf{v}_n(1,1) - \mathbf{v}_{n-1}(1,1)|$$

where $\mathbf{v}_k(1,1)$ represents the approximation of the velocity at the point $(1,1)$ with k gridpoints used for discretization. (The point $(1,1)$ is just an arbitrarily chosen "test point".) The below figure shows how this error varies with n:

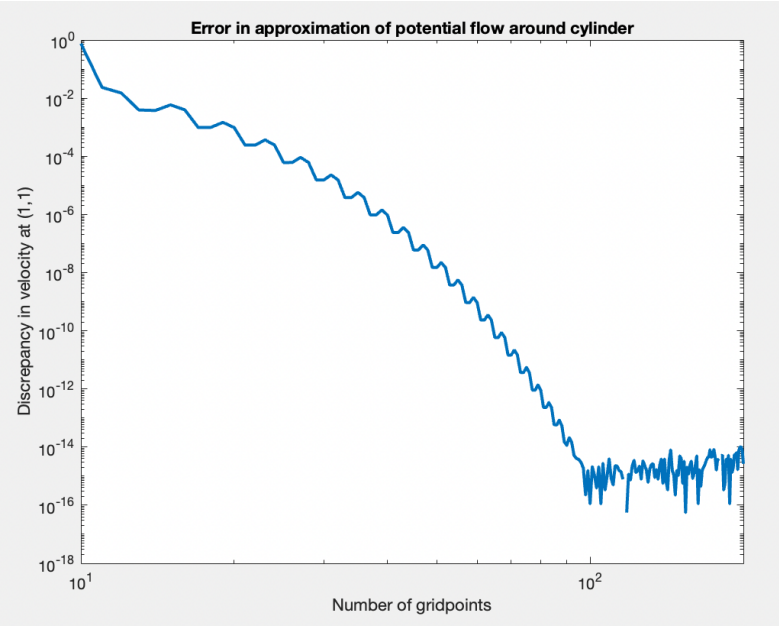


Figure 11: Error decay at the point $(1, 1)$ as the number of gridpoints increases in estimates of potential flow around a circular cylinder.

As we can see from this figure, the error approaches the limits of machine precision very quickly, and the error has already "bottomed out" after only about $n = 100$ grid points.

Now we are ready to test the performance of our approximation on sequences of ellipses with shrinking vertical semiradii. When we shrink the semiradii to very small values, we obtain the following reasonable-looking streamline plot for flow past a plate:

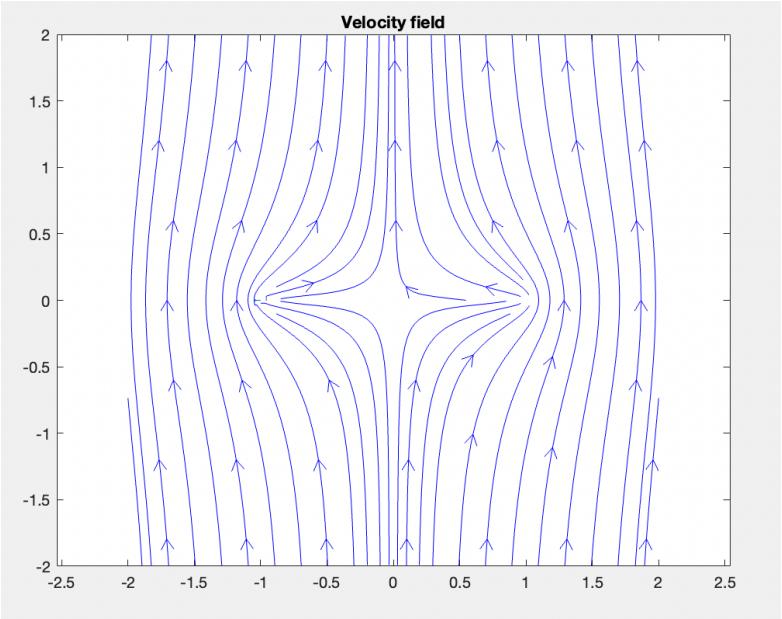


Figure 12: Streamlines for potential flow past a flat plate, approximated by a very thin ellipse.

However, the error plots tell a different story. Below is a plot showing how the velocity approximation at the test point $(1, 1)$ varies with n for the 5 different semiradius values of $s = 10^0, 10^{-1}, ..., 10^{-4}$:

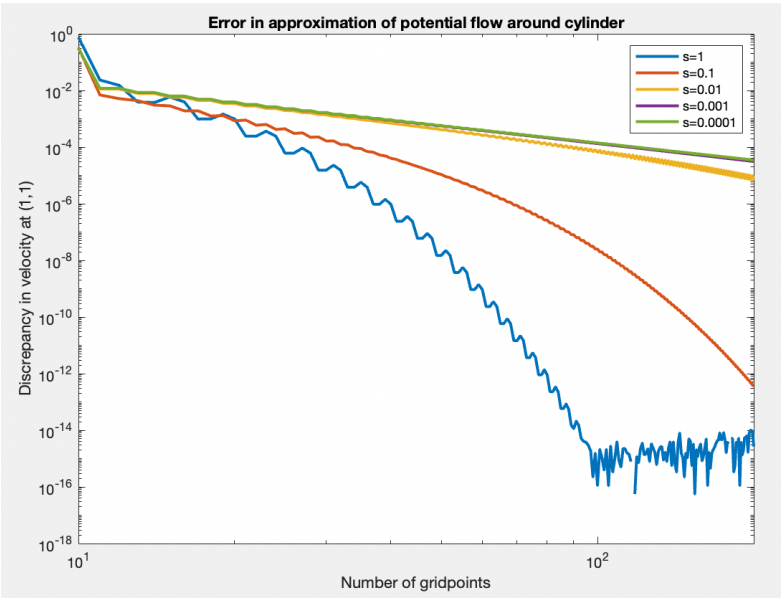


Figure 13: Error decay at the point $(1, 1)$ as the number of gridpoints increases and semiradius decreases in estimates of potential flow around an elliptical cylinder.

For semiradii less than or equal to $s = 0.01$, the error decreases much more slowly compared to the cases of $s = 1$ and $s = 0.1$, and is still at least as large as 10^{-5} when $n = 200$ gridpoints are used - which is sufficient to bring the error close to machine epsilon for both $s = 1$ and $s = 0.1$! Further, calculating flow for values of n much larger than $n = 200$ quickly becomes infeasible because this involves solving a matrix system of size $(n+1) \times (n+1)$, meaning that its time complexity could grow on the order of up to $\mathcal{O}(n^3)$. This means that it would be very difficult to obtain very accurate estimates of the potential flow for values of the semiradius less than or equal to $s = 0.01$.

Recall that all of the above error computations were based on the arbitrarily chosen test point $(1, 1)$ at which we sample the variation in the velocity. However, this is not even a point at which the velocity field is particularly "badly behaved". The most troublesome behavior occurs at the leftmost and rightmost edges of the ellipse which become the endpoints of the approximated plate. Due to this cusp-like geometry, the velocity of the potential flow in the neighborhood of these points is very large, and hence more susceptible to error. If we change our test point to $(1, 0.1)$ instead, which is much closer to the endpoint $(1, 0)$ of the plate, we can see that the error decays much more slowly and erratically:

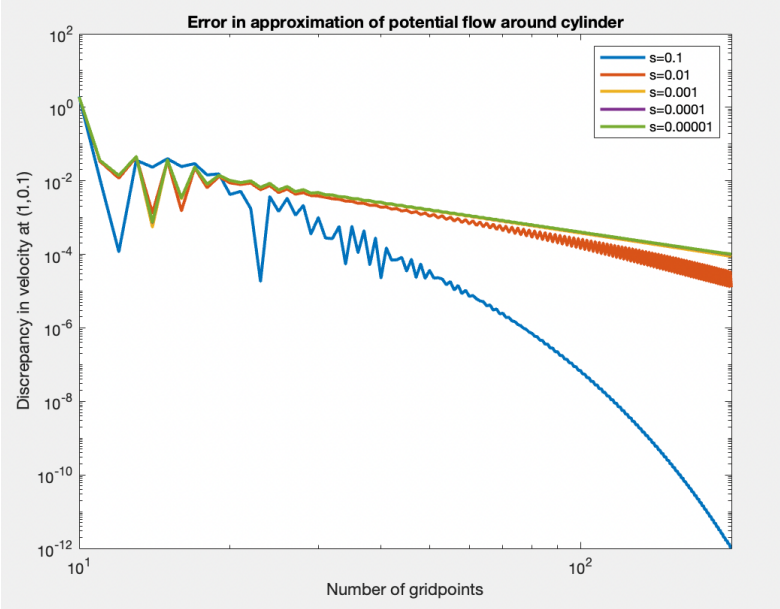


Figure 14: Error decay at the point $(1, 0.1)$ as the number of gridpoints increases in estimates of potential flow around a circular cylinder.

Other "trouble points" occur close to the origin, such as at $(0.1, 0.1)$:

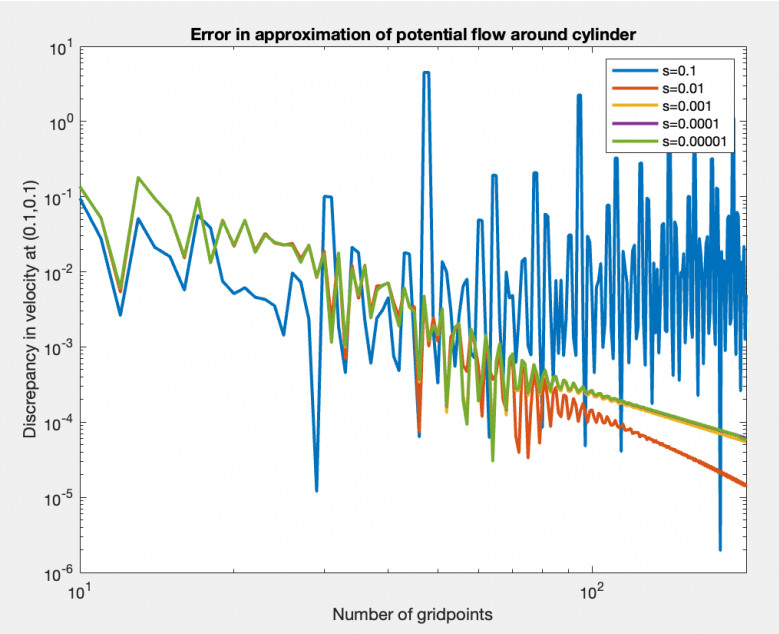


Figure 15: Error decay at the point $(0.1, 0.1)$ as the number of gridpoints increases in estimates of potential flow around a circular cylinder.

4 Conclusion

In the above sections, we have seen how point vortices can be used to discretize the surfaces of objects in order to approximate potential flow around them. We have applied the theoretical equations to implement a function in MATLAB approximating potential flow around the ellipse, and used the limiting behavior of these approximations as $s \to 0$ to estimate fluid flow past a flat plate. As we have seen, it becomes very difficult to obtain accurate approximations as the extremal points of the ellipse display cusplike behavior. This could pose challenges when attempting to calculate potential flow past any object with sharp corners, such as a collection of flat walls or polygons.

The above exploration leaves several questions unanswered, which could be explored further in the future. For instance:

- How can the above methods be refined and optimized to improve the convergence rate of the approximations to the potential flow?
- Can other techniques (perhaps from complex analysis) be used to obtain better approximations of potential flow past objects with sharp corners, or even closed-form analytical expressions for some special cases?
- Why does the error decrease much faster at some points than others relative to the position of the plate? Are approximations necessarily worse

near cusps and stagnation points?

- Do similar problems arise when computing fluid flow according to other models, such as Stokes flow?

For anyone curious who might want to continue experimenting themselves, the MATLAB functions used to compute Γ and plot the streamlines of the induced potential flow are included in the appendix.

5 Acknowledgments

Thanks to my faculty mentor Monika Nitsche for inviting me to get involved in her research, and patiently teaching me a lot about numerical computing and everything I know about fluid dynamics. Much of the background and methodology section of this report is borrowed from the notes that I took during a fluid dynamics crash course is Summer 2021 taught by Professor Nitsche.

Also, many thanks to the UNM department of Arts and Sciences and the ASSURE program for their financial support during the Fall 2021 semester, and to the UNM scholarship office for their amazing National Merit Scholarship, without which I would be unable to study at this great university!

References

[1] A. J. Chorin and J. E. Marsden, *A mathematical introduction to fluid mechanics*. Texts in applied mathematics, Springer-Verlag.

[2] M. Nitsche, "Minicourse 4: An introduction to fluid dynamics."

[3] M. Nitsche, "Evaluation of near-singular integrals with application to vortex sheet flow."

A Appendix A: Various point vortices

The following pages consist of a short report that explores the behavior of various different finite collections of point vortices and their induced velocity fields.

Table of Contents

Single vortex streamfunction

```
function psi = vortex_psi(cx, cy, speed_int, xg, yg)
    radii = sqrt((xg-cx).^2+(yg-cy).^2);
    psi = -speed_int(radii);
end
```

In the above code, `speed_int` represents an antiderivative of the speed of the vortex as a function of the distance from its center. We will consider two types of vortices: those in which the speed is constant everywhere, which we will call **v-constant** vortices, and those in which the speed is inversely proportional to the distance from the center, which we will call **v-slow** vortices. To find the streamfunction for a flow containing two or more such vortices, we just need to sum the streamfunctions for each of the individual vertices considered independently. For flows with multiples vortices, we can use the above simple function to calculate the streamfunction for each vortex, and then add them together.

Two-vortex flows

What do streamlines look like when we have two different vortices? There are multiple possibilities to consider:

- Both vortices could be v-constant, or both could be v-slow, or there could be one of each type
- The vortices could both be rotating clockwise or counterclockwise, or be rotating with opposite orientations
- The vortices could have varying magnitudes of rotation
- The distance between the centers of the vortices can vary

We will consider several different combinations of these possibilities.

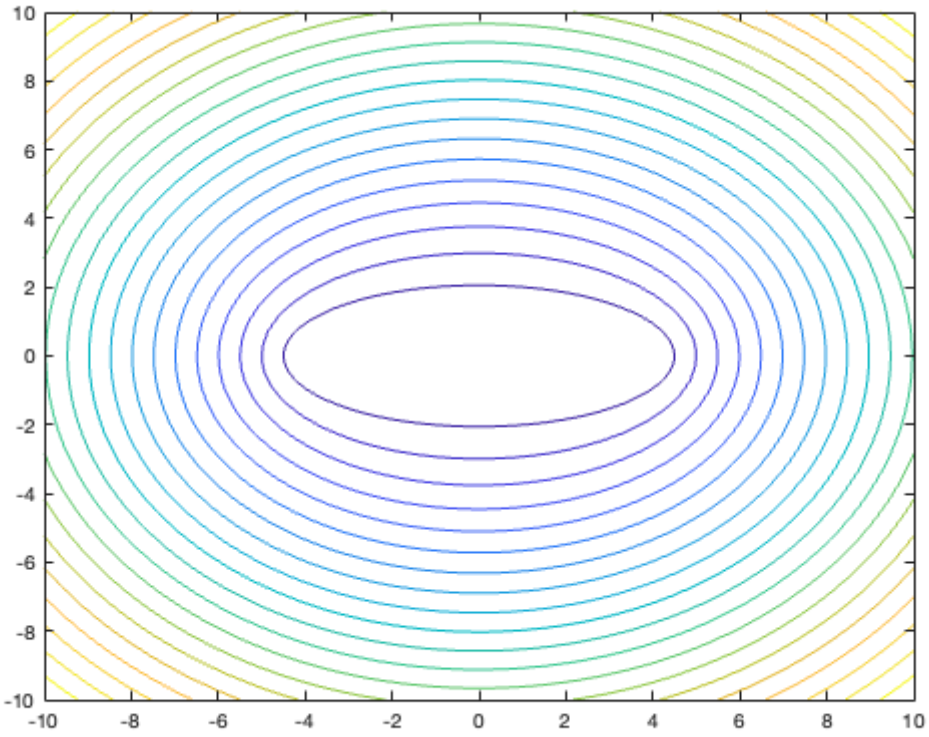
Example 1. Two v-constant vortices with the same orientation and speed:

```
x = linspace(-10,10);
y = linspace(-10,10);
[xg,yg] = meshgrid(x,y);
```



```
Speed = inline('-r');
```

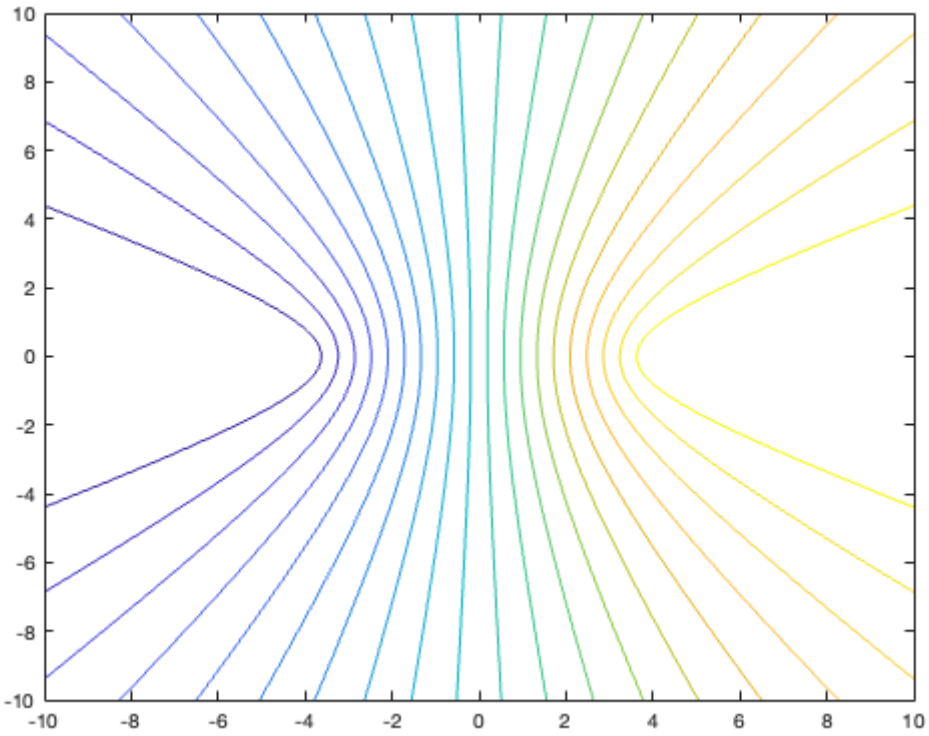
```
psi1 = vortex_psi(-4,0,Speed,xg,yg);
psi2 = vortex_psi(4,0,Speed,xg,yg);
psi = psi1+psi2;

contour(xg,yg,psi,20);
```



Note that, on the line segment joining the centers of the vortices, velocity equals zero, because the contributions of the rotations from the two vortices cancel each other. The streamlines appear to be ellipses.

Example 2. Two v-constant vortices with the opposite orientation and same speed:

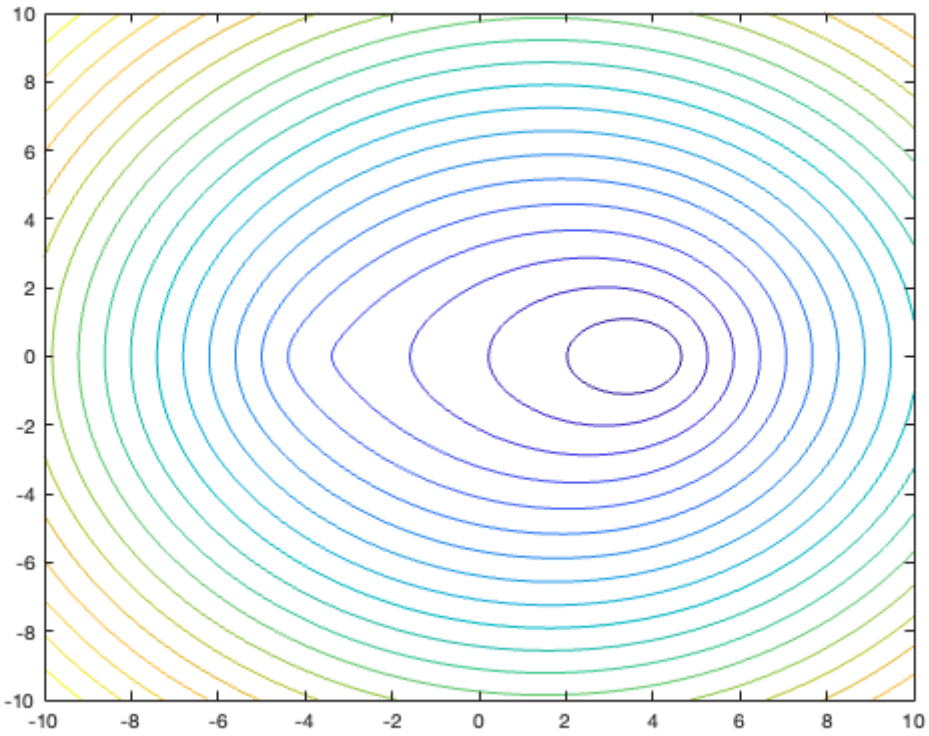
```
Speed1 = inline('-r'); % counterclockwise
Speed2 = inline('r');  % clockwise
psi1 = vortex_psi(-4,0,Speed1,xg,yg);
psi2 = vortex_psi(4,0,Speed2,xg,yg);
psi = psi1+psi2;

contour(xg,yg,psi,20);
```



Remarkably, this flow has no closed streamlines, even though it is a combination of two vortices! In this case, the streamlines appear to be hyperbolas.

Example 3. Two v-constant vortices with the same orientation, in which one is twice as strong as the other:

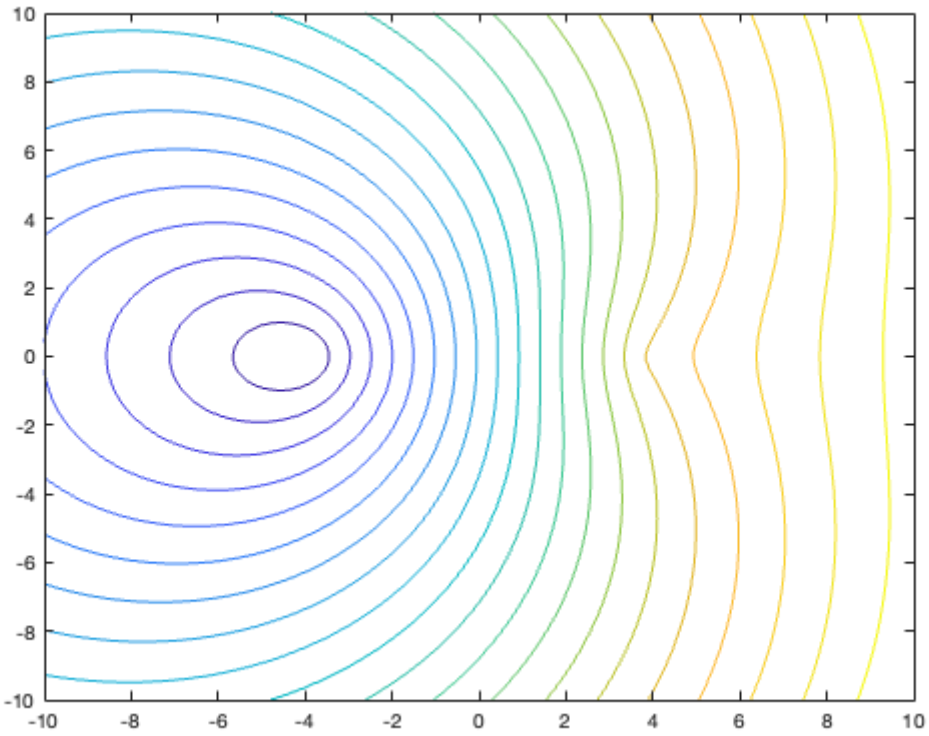
```
Speed1 = inline('-r');   % weaker vortex
Speed2 = inline('-2*r'); % stronger vortex
psi1 = vortex_psi(-4,0,Speed1,xg,yg);
psi2 = vortex_psi(4,0,Speed2,xg,yg);
psi = psi1+psi2;

contour(xg,yg,psi,20);
```



This time, the streamlines are closed and roughly elliptical in shape near the center of the stronger vortex, but the larger streamlines are deformed slightly by the weaker vortex, making them egg-shaped. No closed streamlines at all surround the weak vortex, but it does seem to cause a sharp/pointy "cusp" in the streamlines that pass near its center.

Example 4. Two v-constant vortices with different orientations, in which one is twice as strong as the other:

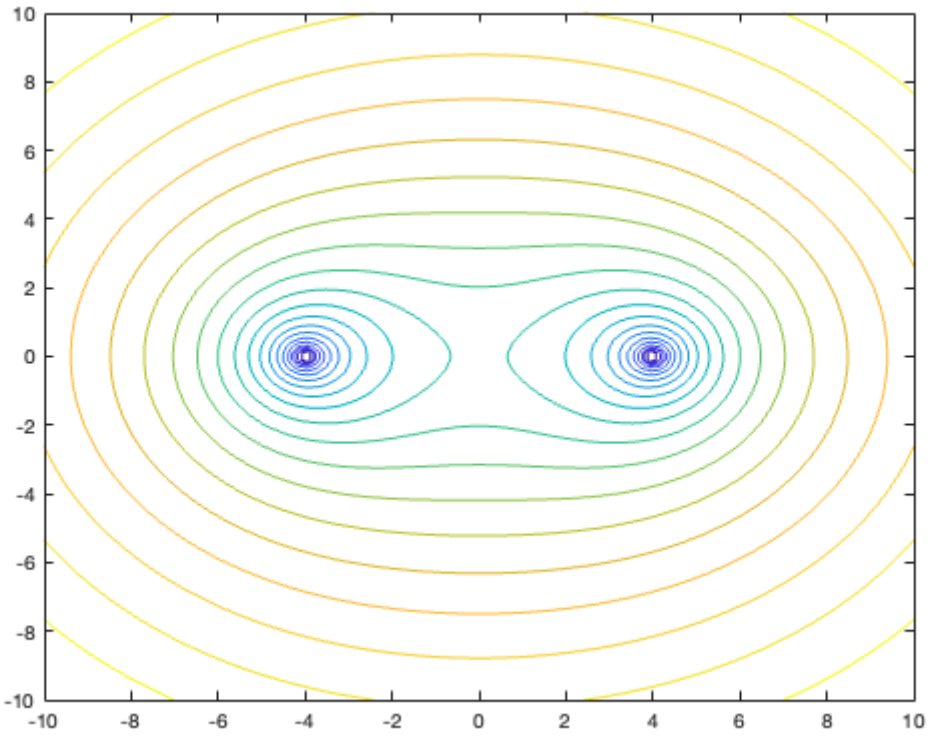
```
Speed1 = inline('-2*r'); % stronger vortex, counterclockwise
Speed2 = inline('r');    % weaker vortex, clockwise
psi1 = vortex_psi(-4,0,Speed1,xg,yg);
psi2 = vortex_psi(4,0,Speed2,xg,yg);
psi = psi1+psi2;

contour(xg,yg,psi,20);
```



Yet again, only the stronger vortex has closed streamlines surrounding it, and the weak vortex only manages to deform the larger streamlines that come near it. Now let's consider examples in which both vortices are v-slow instead of v-constant.

Example 5. Two v-slow vortices with the same orientation and strength:

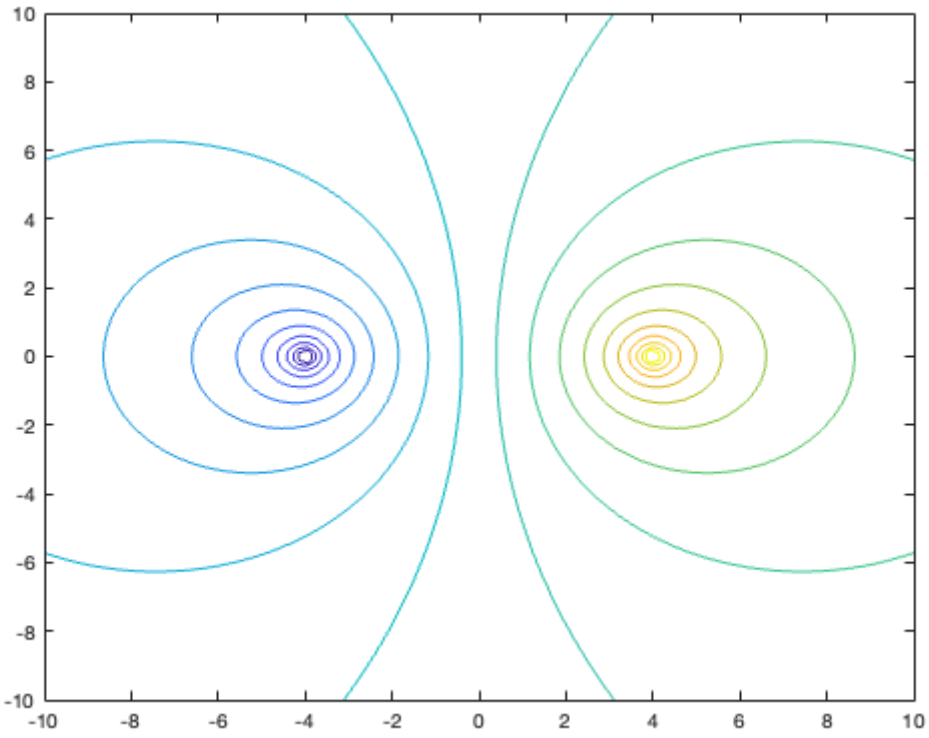
```
Speed = inline('-log(r)/(4*pi)');
psi1 = vortex_psi(-4,0,Speed,xg,yg);
psi2 = vortex_psi(4,0,Speed,xg,yg);
psi = psi1+psi2;

contour(xg,yg,psi,20);
```



This is similar to the case with two v-constant vortices in that the larger streamlines look somewhat like ellipses, but in this case, each vortex has "its own" closed paths surrounding it. This is because the velocity contribution of each vortex shrinks with distance in the v-slow case, meaning that each vortex dominates trajectories that start close to it. Note that there is a stationary point at the origin.

Example 6. Two v-slow vortices with opposite orientations and the same strength:

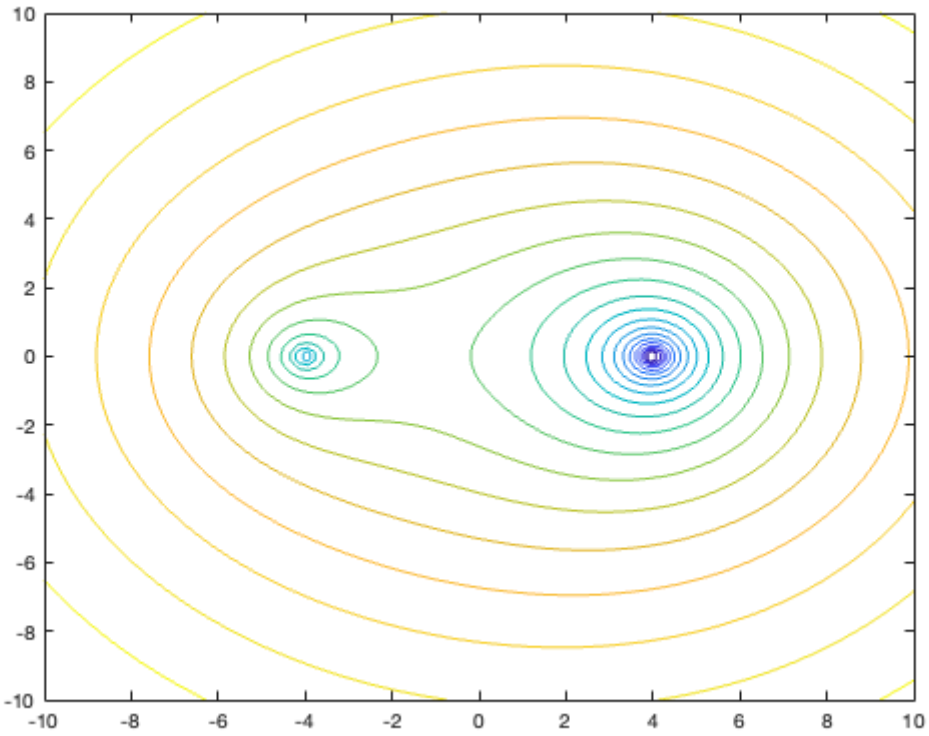
```
Speed1 = inline('-log(r)/(4*pi)');  % counterclockwise
Speed2 = inline('log(r)/(4*pi)');   % clockwise
psi1 = vortex_psi(-4,0,Speed1,xg,yg);
psi2 = vortex_psi(4,0,Speed2,xg,yg);
psi = psi1+psi2;

contour(xg,yg,psi,20);
```



In this case, we have the similar phenomenon of each vortex center being surrounded by small closed streamlines, but there are no larger streamlines that enclose both vortex centers. Instead, there is a straight-line trajectory lying entirely on the y-axis that separates the "territory" of one vortex from the other.

Example 7. Two v-slow vortices with the same orientation, in which one is twice as strong as the other:

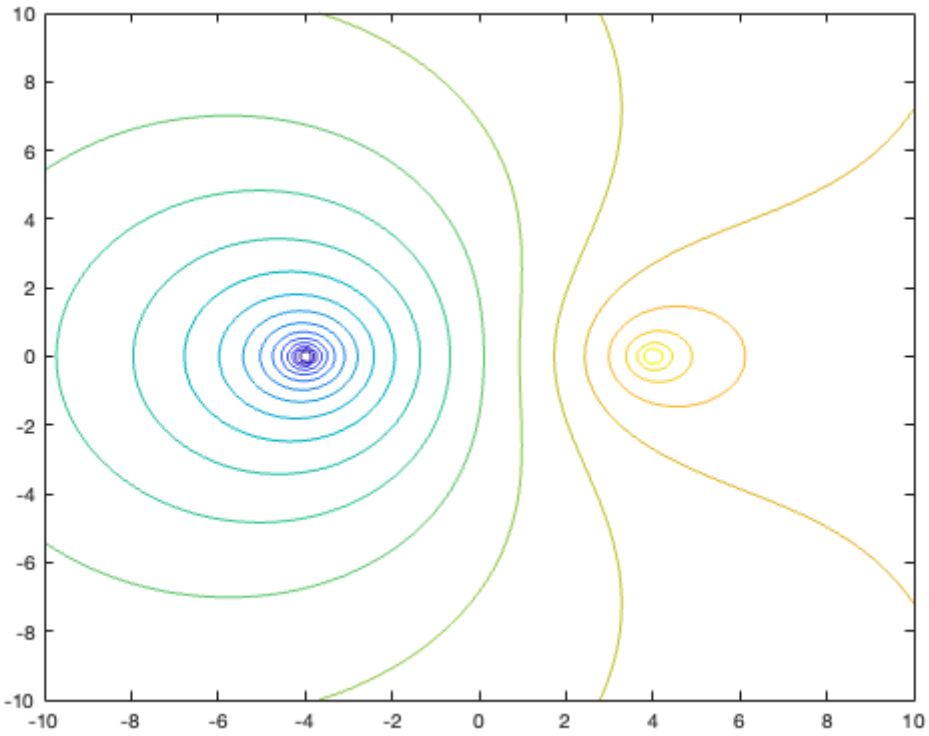
```
Speed1 = inline('-log(r)/(4*pi)'); % weaker vortex
Speed2 = inline('-log(r)/(2*pi)'); % stronger vortex
psi1 = vortex_psi(-4,0,Speed1,xg,yg);
psi2 = vortex_psi(4,0,Speed2,xg,yg);
psi = psi1+psi2;

contour(xg,yg,psi,20);
```



Here we have something similar to Example 5, but the stronger vortex has more/larger closed streamlines surrounding its center. This makes sense, since it "dominates" a larger area by virtue of its increased strength.

Example 8. Two v-slow vortices with opposite orientations, in which one is twice as strong as the other:

```
Speed1 = inline('-log(r)/(2*pi)'); % stronger vortex, counterclockwise
Speed2 = inline('log(r)/(4*pi)');  % weaker vortex, clockwise
psi1 = vortex_psi(-4,0,Speed1,xg,yg);
psi2 = vortex_psi(4,0,Speed2,xg,yg);
psi = psi1+psi2;

contour(xg,yg,psi,20);
```

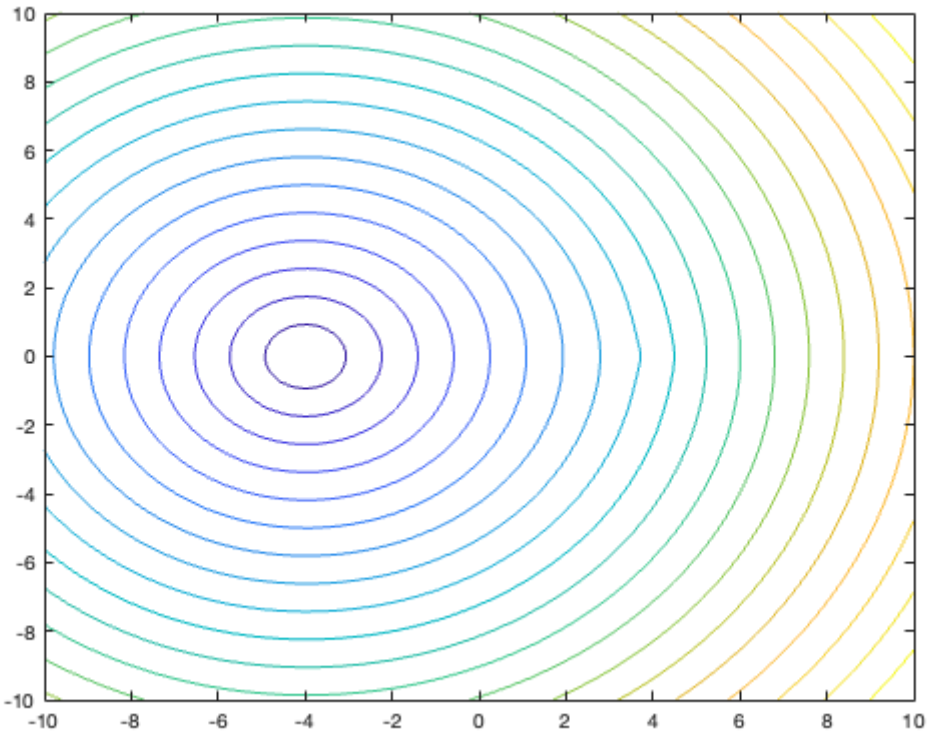


This time we have an analog of Example 6, again deformed in favor of the strong vortex. There are no trajectories enclosing the centers of both vortices, and there again appears to be a (non-closed) trajectory separating their "territories", but it is not a straight line anymore.

Now let's consider some two-vortex flows with one v-constant and one v-slow vortex.

Example 9. One v-constant and one v-slow vortex with the same orientation.

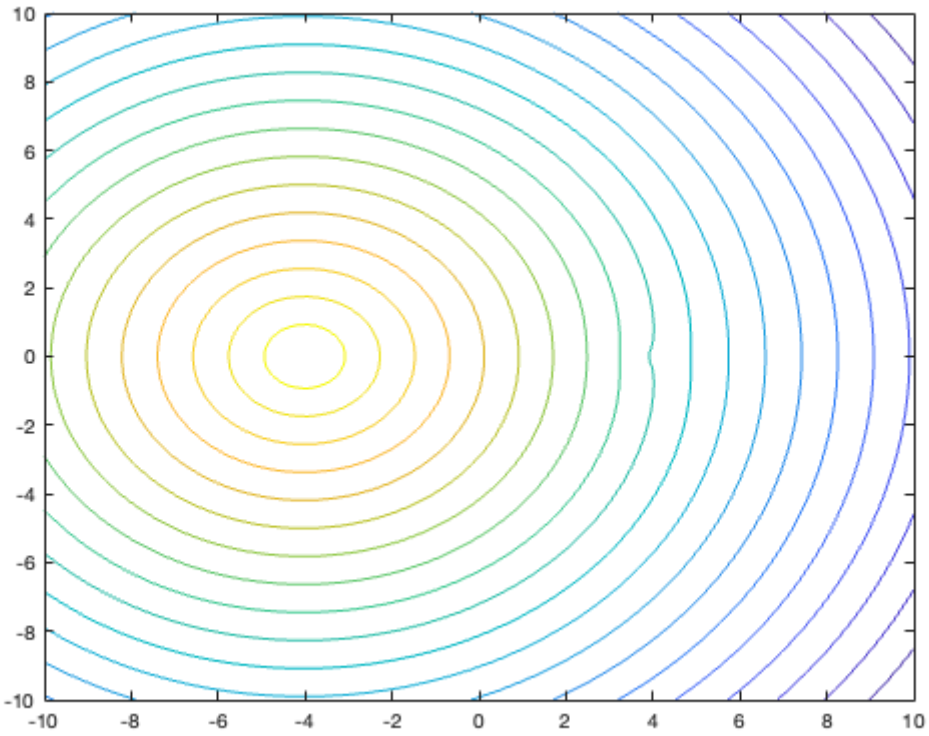
```
Speed1 = inline('-r');              % v-constant vortex
Speed2 = inline('-log(r)/(4*pi)'); % v-slow vortex
psi1 = vortex_psi(-4,0,Speed1,xg,yg);
psi2 = vortex_psi(4,0,Speed2,xg,yg);
psi = psi1+psi2;

contour(xg,yg,psi,20);
```



As we would expect from two vortices that are "working together" (i.e. turning in the same direction), the larger streamlines enclose both centers and look almost elliptical (even vaguely circular, for the larger streamlines). Although it isn't visible from this plot, the v-slow vortex has tiny closed streamlines surrounding it.

Example 10. One v-constant and one v-slow vortex, with opposite orientations.

```
Speed1 = inline('r');              % v-constant vortex, clockwise
Speed2 = inline('-log(r)/(4*pi)'); % v-slow vortex, counterclockwise
psi1 = vortex_psi(-4,0,Speed1,xg,yg);
psi2 = vortex_psi(4,0,Speed2,xg,yg);
psi = psi1+psi2;

contour(xg,yg,psi,20);
```



Again, the v-slow vortex only has tiny closed streamlines surrounding it (invisible in this plot), and the v-constant dominates the flow everywhere else, being only slightly deformed by the v-slow vortex.

Three or more v-constant vortices

We won't look at these cases as thoroughly, because there are many more of them to consider, but here is some code that (non-exhaustively) covers some of the possible combinations of three v-constant vortices:

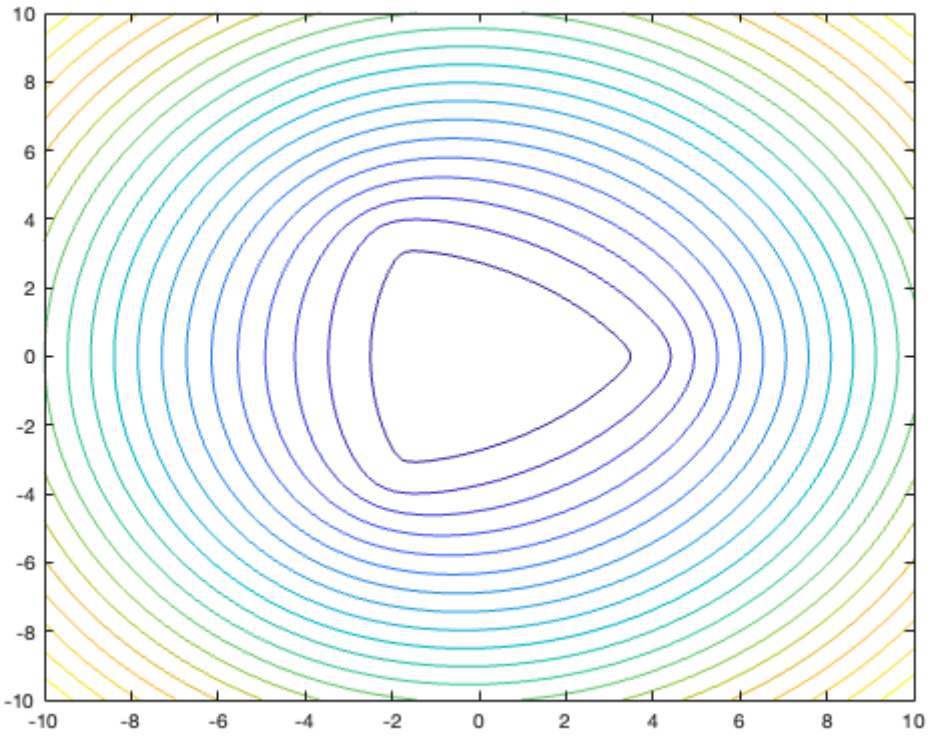
```
Speed1 = inline('-r');   % counterclockwise weak flow
Speed2 = inline('r');    % clockwise weak flow
Speed3 = inline('-2*r'); % counterclockwise strong flow
Speed4 = inline('2*r');  % clockwise strong flow

x1 = 4; y1 = 0;
x2 = 4*cos(2*pi/3); y2 = 4*sin(2*pi/3);
x3 = 4*cos(4*pi/3); y3 = 4*sin(4*pi/3);
```

Example 11. Three v-constant vortices, same strength, same orientation.

```
psi1 = vortex_psi(x1,y1,Speed1,xg,yg);
psi2 = vortex_psi(x2,y2,Speed1,xg,yg);
psi3 = vortex_psi(x3,y3,Speed1,xg,yg);
psi = psi1+psi2+psi3;
```

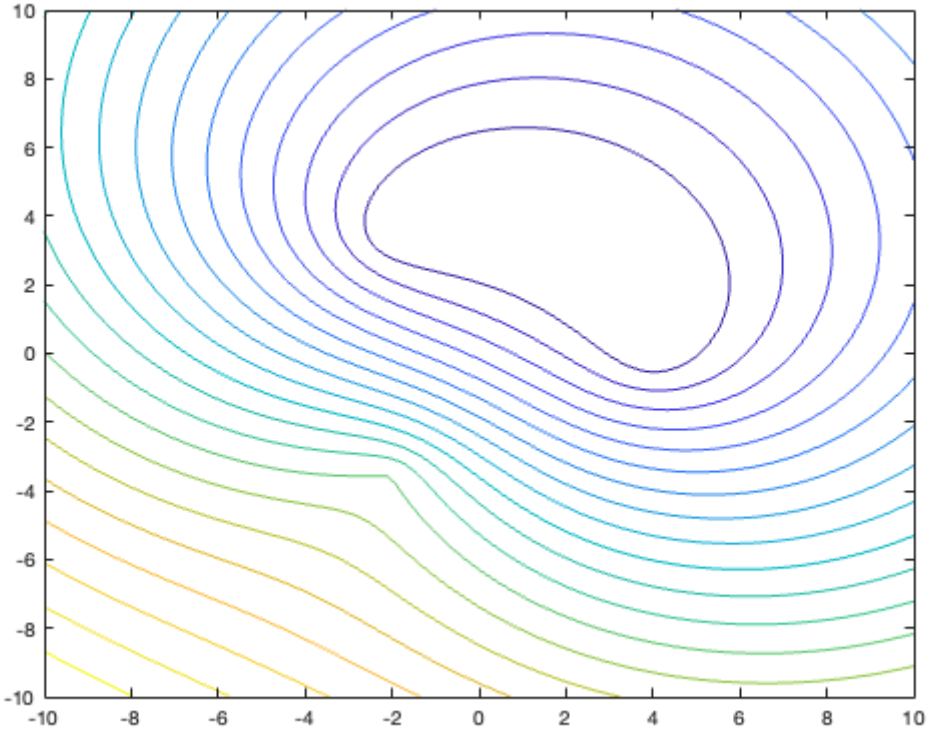


```
contour(xg,yg,psi,20)
```



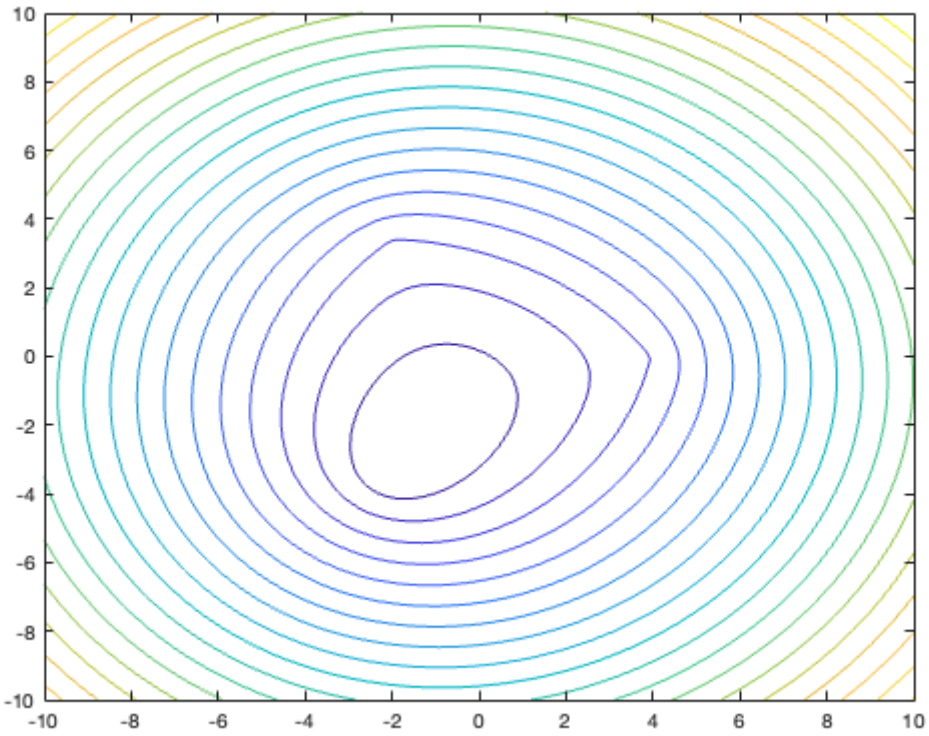
Example 12. Three v-constant vortices, same strength, two with the same orientation, one with the opposite orientation.

```
psi1 = vortex_psi(x1,y1,Speed1,xg,yg);
psi2 = vortex_psi(x2,y2,Speed1,xg,yg);
psi3 = vortex_psi(x3,y3,Speed2,xg,yg);
psi = psi1+psi2+psi3;

contour(xg,yg,psi,20)
```



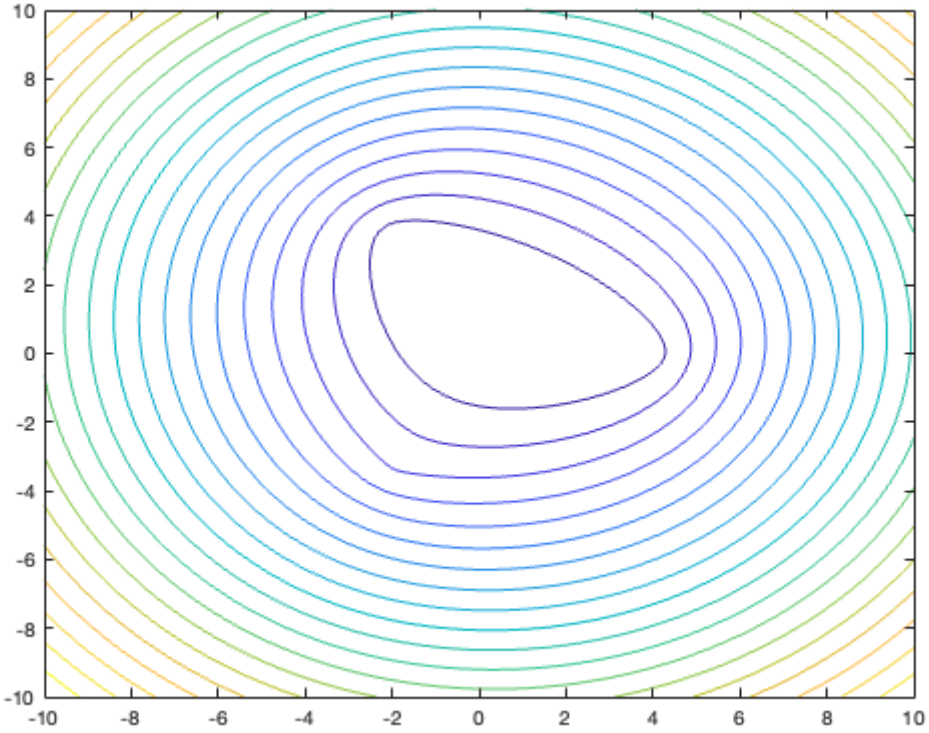
Example 13. Three v-constant vortices, same orientation, one stronger than the other two.

```
psi1 = vortex_psi(x1,y1,Speed1,xg,yg);
psi2 = vortex_psi(x2,y2,Speed1,xg,yg);
psi3 = vortex_psi(x3,y3,Speed3,xg,yg);
psi = psi1+psi2+psi3;

contour(xg,yg,psi,20)
```



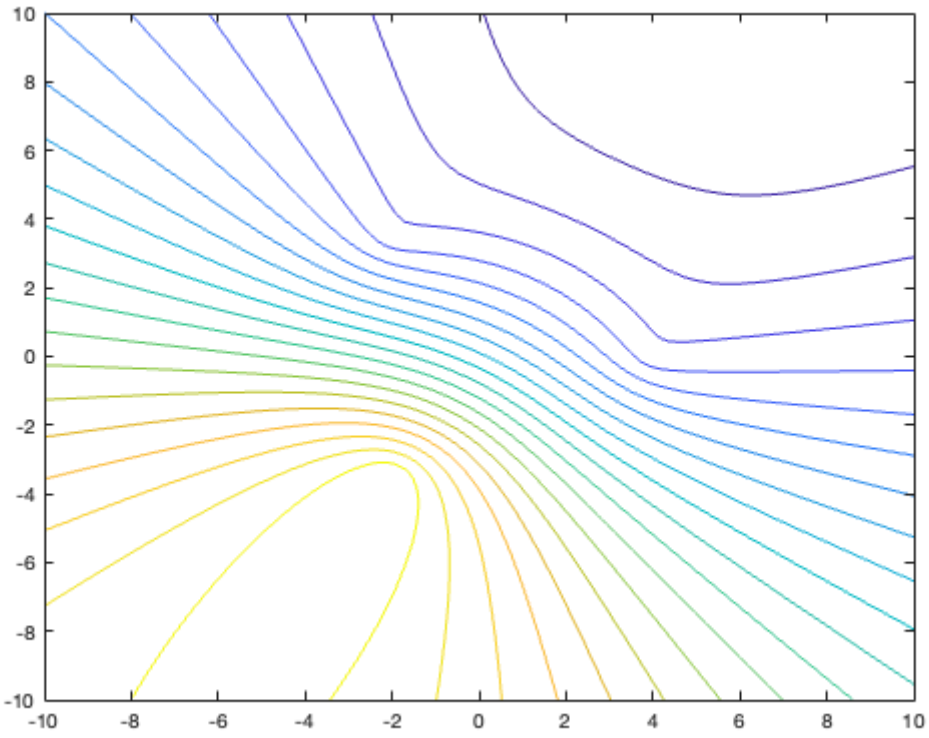
Example 14. Three v-constant vortices, same orientation, one weaker than the other two.

```
psi1 = vortex_psi(x1,y1,Speed3,xg,yg);
psi2 = vortex_psi(x2,y2,Speed3,xg,yg);
psi3 = vortex_psi(x3,y3,Speed1,xg,yg);
psi = psi1+psi2+psi3;

contour(xg,yg,psi,20)
```



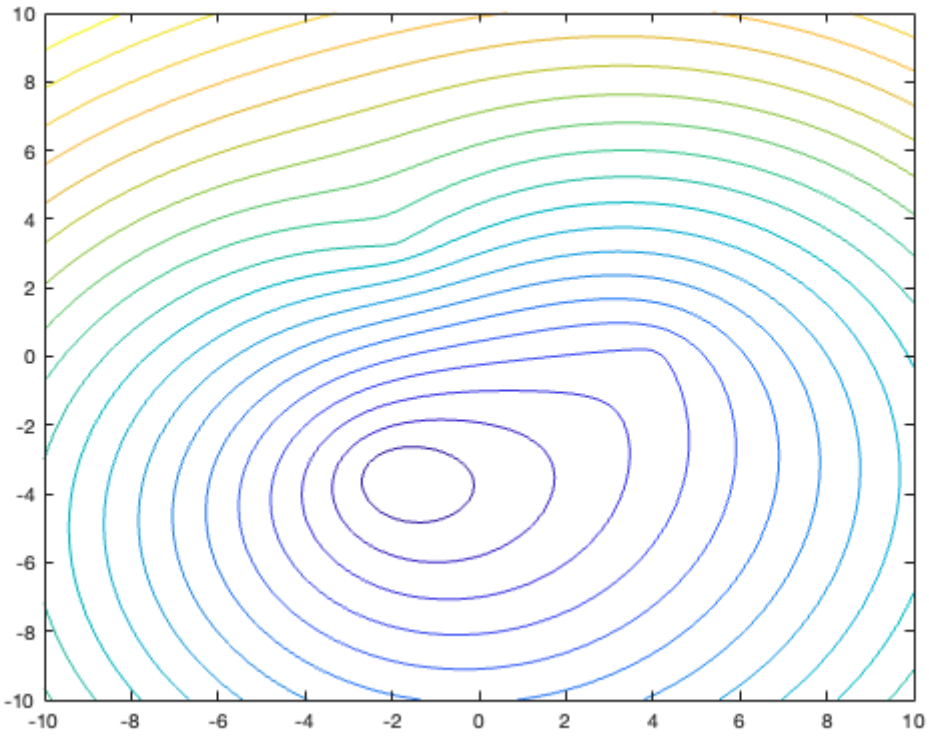
Example 15. Three v-constant vortices, one with the opposite orientation and stronger than the other two.

```
psi1 = vortex_psi(x1,y1,Speed1,xg,yg);
psi2 = vortex_psi(x2,y2,Speed1,xg,yg);
psi3 = vortex_psi(x3,y3,Speed4,xg,yg);
psi = psi1+psi2+psi3;

contour(xg,yg,psi,20)
```



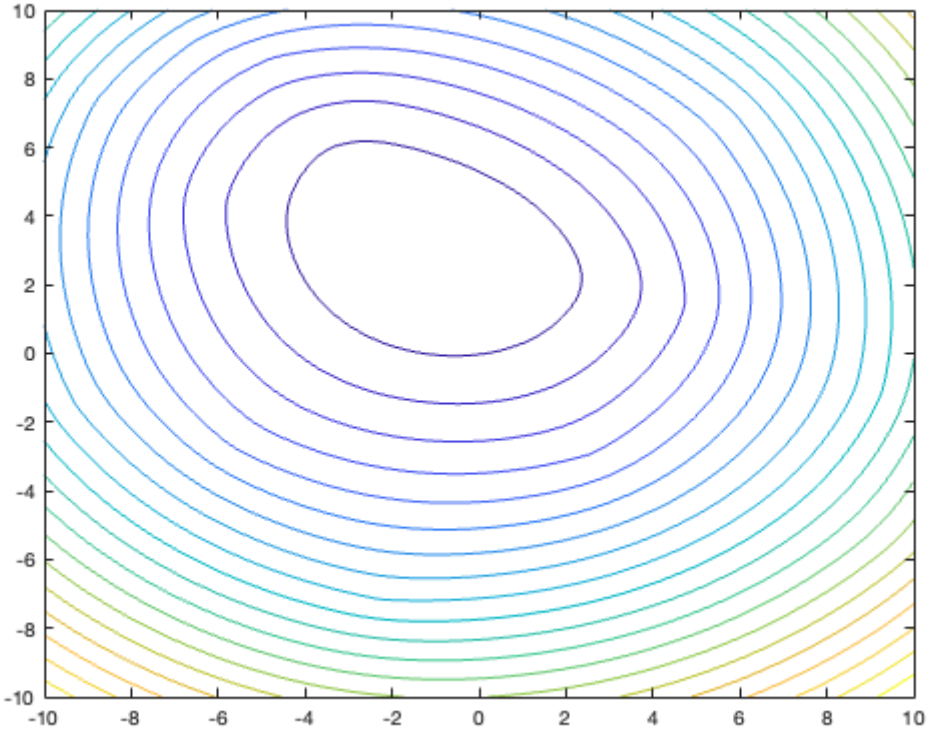
Example 16. Three v-constant vortices, one with the opposite orientation than the other two, and a different one stronger than the other two.

```
psi1 = vortex_psi(x1,y1,Speed1,xg,yg);
psi2 = vortex_psi(x2,y2,Speed2,xg,yg);
psi3 = vortex_psi(x3,y3,Speed3,xg,yg);
psi = psi1+psi2+psi3;

contour(xg,yg,psi,20)
```



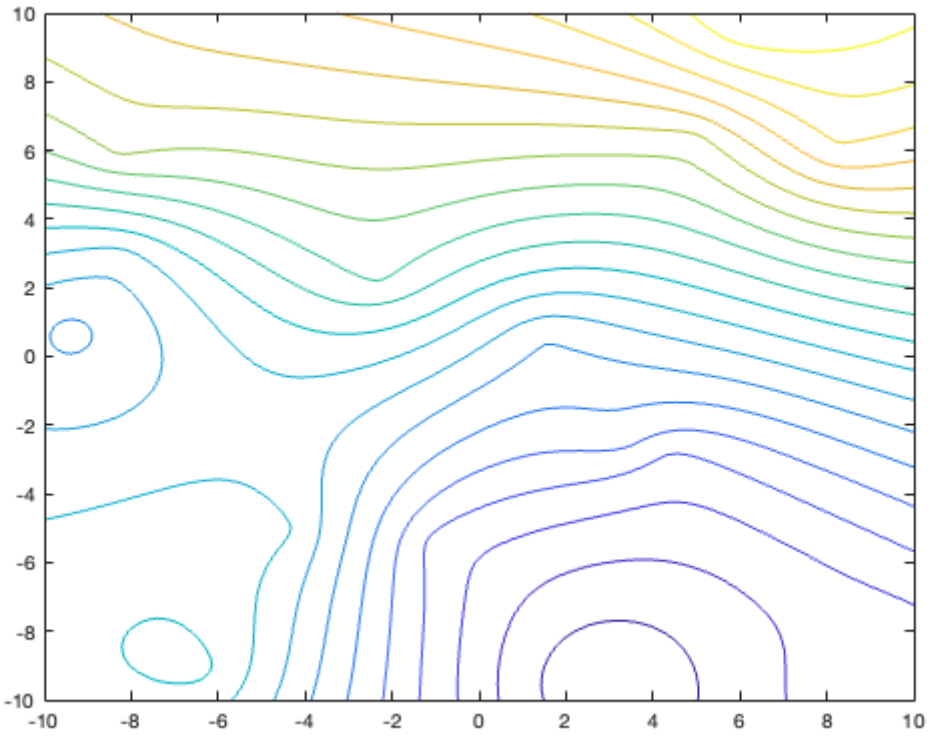
Example 17. 16 randomly placed v-constant vortices, all with the same orientation.

```
psi = 0;
for i=1:16
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed1,xg,yg);
end

contour(xg,yg,psi,20)
```



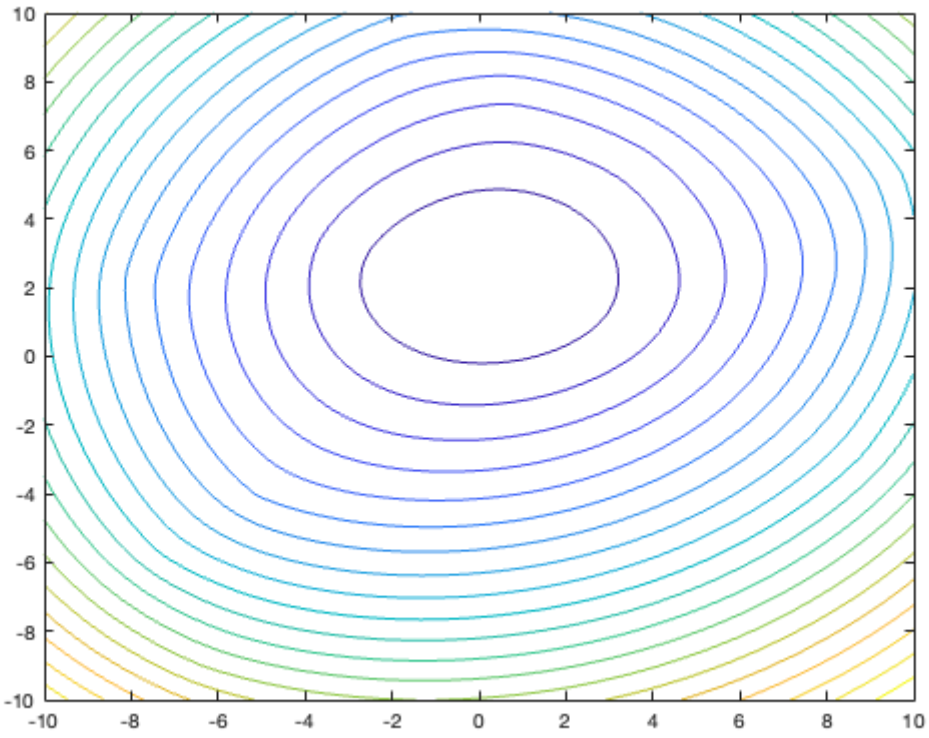
Example 18. 16 randomly placed v-constant vortices, half of which have the opposite orientation of the other half.

```
psi = 0;
for i=1:8
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed1,xg,yg);
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed2,xg,yg);
end

contour(xg,yg,psi,20)
```



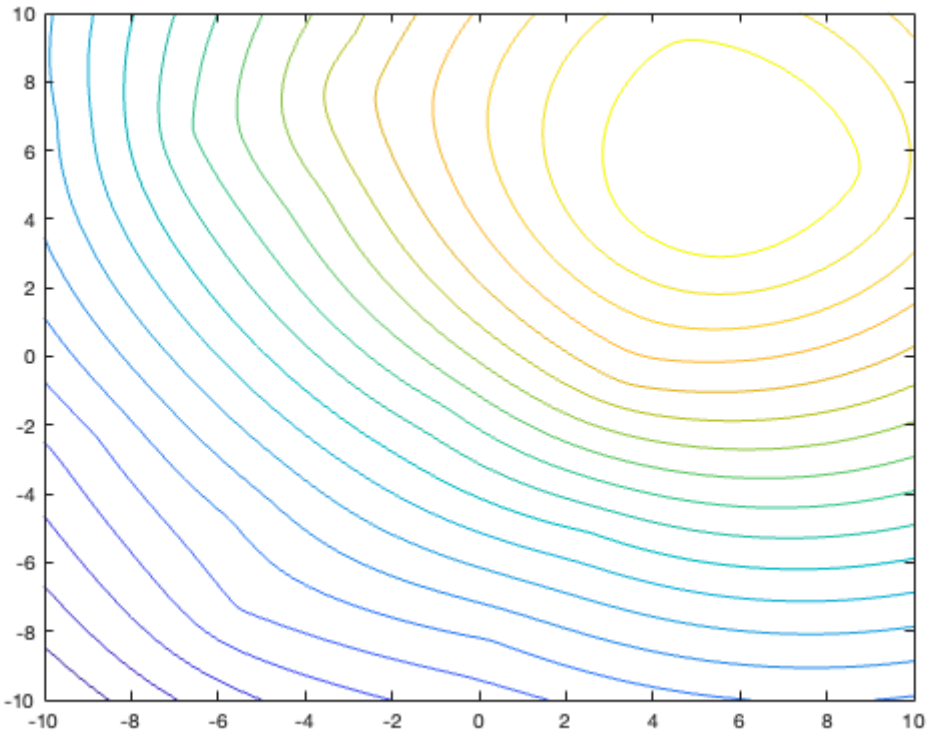
Example 19. 16 randomly placed v-constant vortices, half of which are stronger and half of which are weaker.

```
psi = 0;
for i=1:8
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed1,xg,yg);
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed3,xg,yg);
end

contour(xg,yg,psi,20)
```



Example 20. 16 randomly placed v-constant vortices, half of which are stronger than the others and have the opposite orientation.

```
psi = 0;
for i=1:8
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed1,xg,yg);
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed4,xg,yg);
end

contour(xg,yg,psi,20)
```

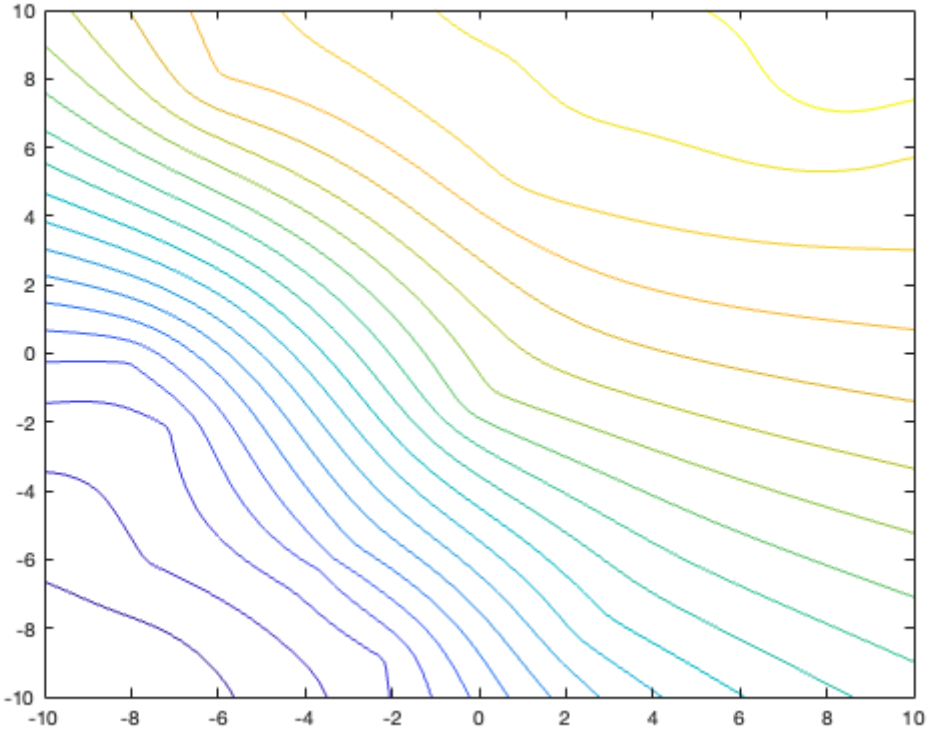


Example 21. 16 randomly placed v-constant vortices, with one-fourth being counterclockwise and weak, one-fourth being counterclockwise and strong, one-fourth being clockwise and weak, and one-fourth being clockwise and strong.

```
psi = 0;
for i=1:4
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed1,xg,yg);
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed2,xg,yg);
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed3,xg,yg);
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed4,xg,yg);
end

contour(xg,yg,psi,20)
```

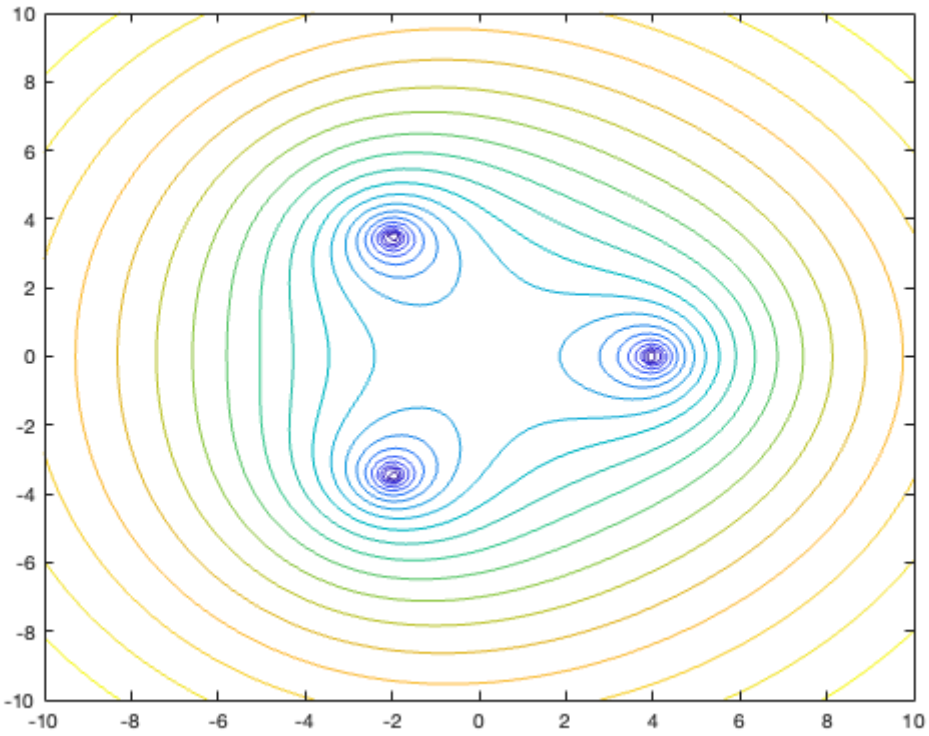
Three or more v-slow vortices

```
Speed1 = inline('-log(r)/(4*pi)'); % counterclockwise weak flow
Speed2 = inline('log(r)/(4*pi)');  % clockwise weak flow
Speed3 = inline('-log(r)/(2*pi)'); % counterclockwise strong flow
Speed4 = inline('log(r)/(2*pi)');  % clockwise strong flow
```



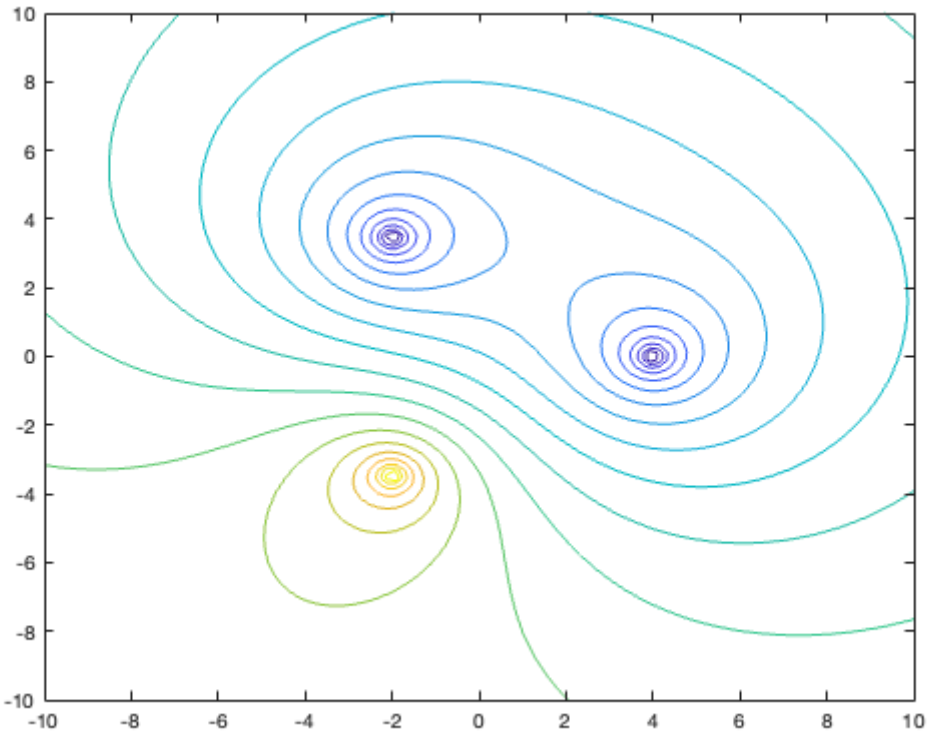
Example 22. Three v-slow vortices, same strength, same orientation.

```
psi1 = vortex_psi(x1,y1,Speed1,xg,yg);
psi2 = vortex_psi(x2,y2,Speed1,xg,yg);
psi3 = vortex_psi(x3,y3,Speed1,xg,yg);
psi = psi1+psi2+psi3;

contour(xg,yg,psi,20)
```



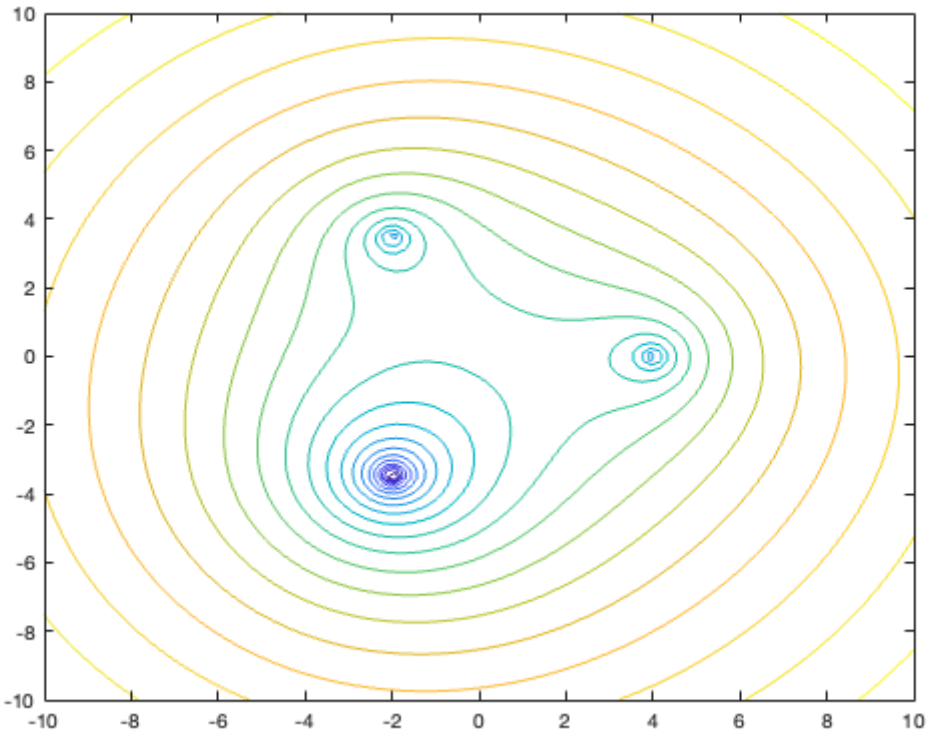
Example 23. Three v-slow vortices, same strength, two with the same orientation, one with the opposite orientation.

```
psi1 = vortex_psi(x1,y1,Speed1,xg,yg);
psi2 = vortex_psi(x2,y2,Speed1,xg,yg);
psi3 = vortex_psi(x3,y3,Speed2,xg,yg);
psi = psi1+psi2+psi3;

contour(xg,yg,psi,20)
```



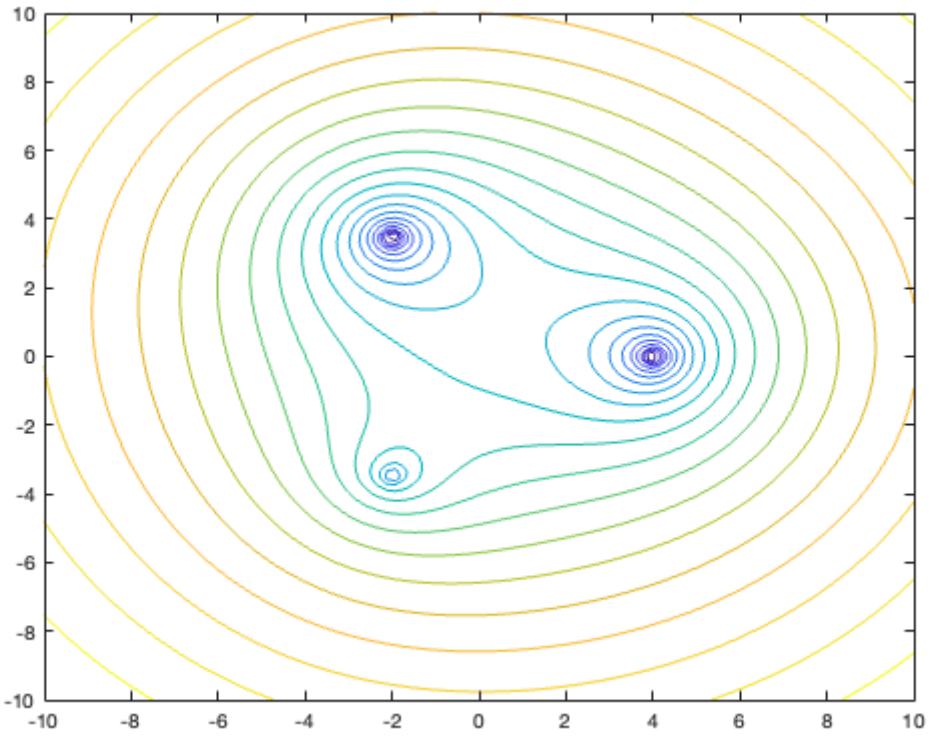
Example 24. Three v-slow vortices, same orientation, one stronger than the other two.

```
psi1 = vortex_psi(x1,y1,Speed1,xg,yg);
psi2 = vortex_psi(x2,y2,Speed1,xg,yg);
psi3 = vortex_psi(x3,y3,Speed3,xg,yg);
psi = psi1+psi2+psi3;

contour(xg,yg,psi,20)
```



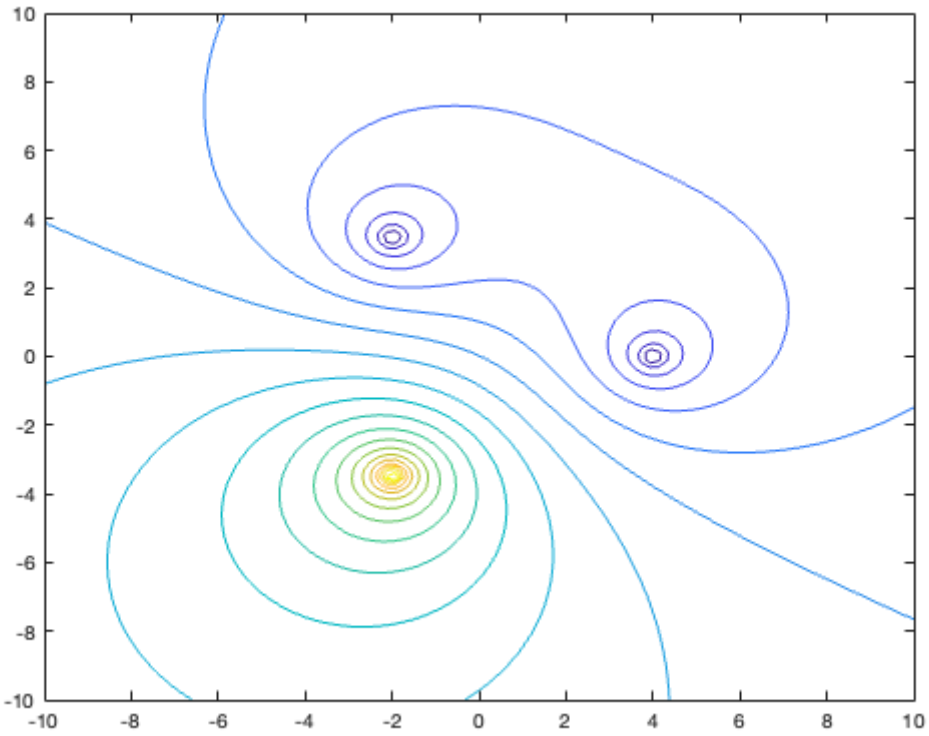
Example 25. Three v-slow vortices, same orientation, one weaker than the other two.

```
psi1 = vortex_psi(x1,y1,Speed3,xg,yg);
psi2 = vortex_psi(x2,y2,Speed3,xg,yg);
psi3 = vortex_psi(x3,y3,Speed1,xg,yg);
psi = psi1+psi2+psi3;

contour(xg,yg,psi,20)
```



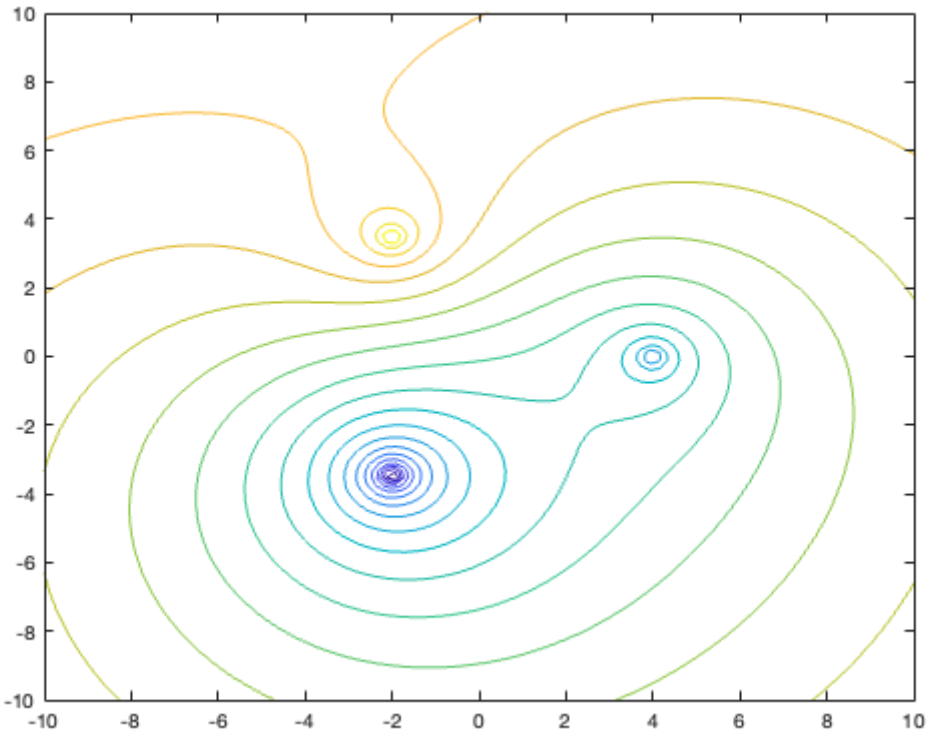
Example 26. Three v-slow vortices, one with the opposite orientation and stronger than the other two.

```
psi1 = vortex_psi(x1,y1,Speed1,xg,yg);
psi2 = vortex_psi(x2,y2,Speed1,xg,yg);
psi3 = vortex_psi(x3,y3,Speed4,xg,yg);
psi = psi1+psi2+psi3;

contour(xg,yg,psi,20)
```



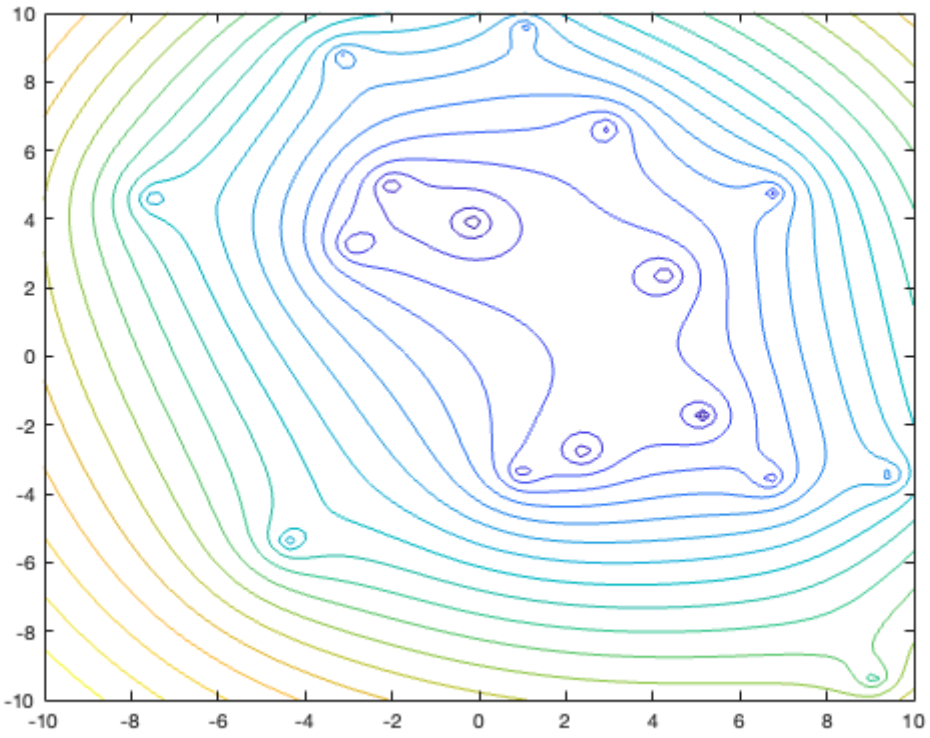
Example 27. Three v-slow vortices, one with the opposite orientation than the other two, and a different one stronger than the other two.

```
psi1 = vortex_psi(x1,y1,Speed1,xg,yg);
psi2 = vortex_psi(x2,y2,Speed2,xg,yg);
psi3 = vortex_psi(x3,y3,Speed3,xg,yg);
psi = psi1+psi2+psi3;

contour(xg,yg,psi,20)
```



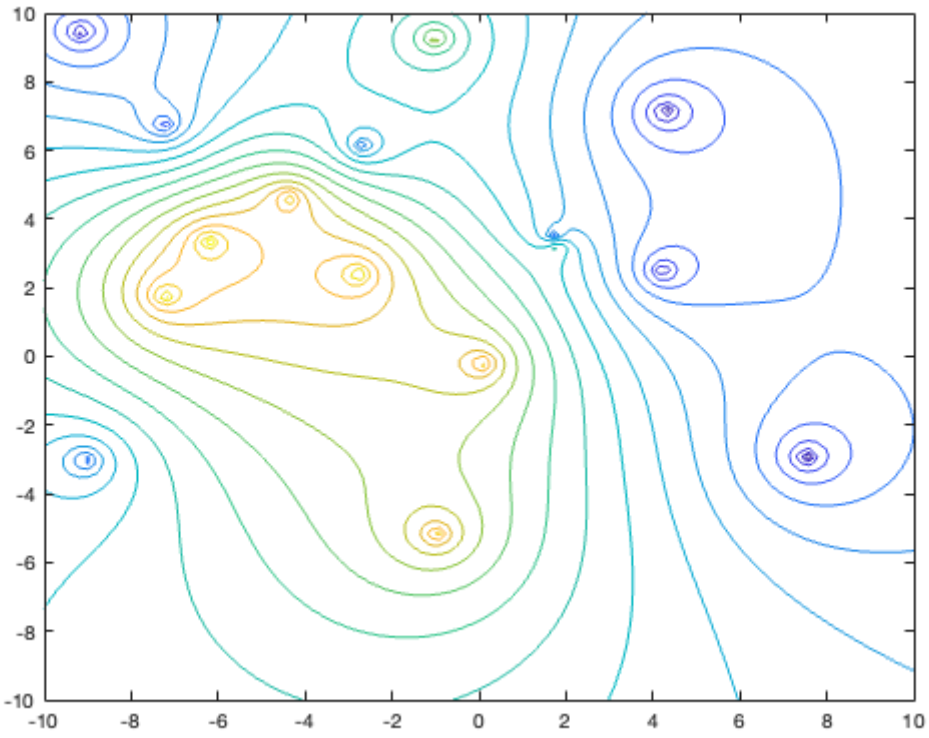
Example 28. 16 randomly placed v-slow vortices, all with the same orientation.

```
psi = 0;
for i=1:16
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed1,xg,yg);
end

contour(xg,yg,psi,20)
```



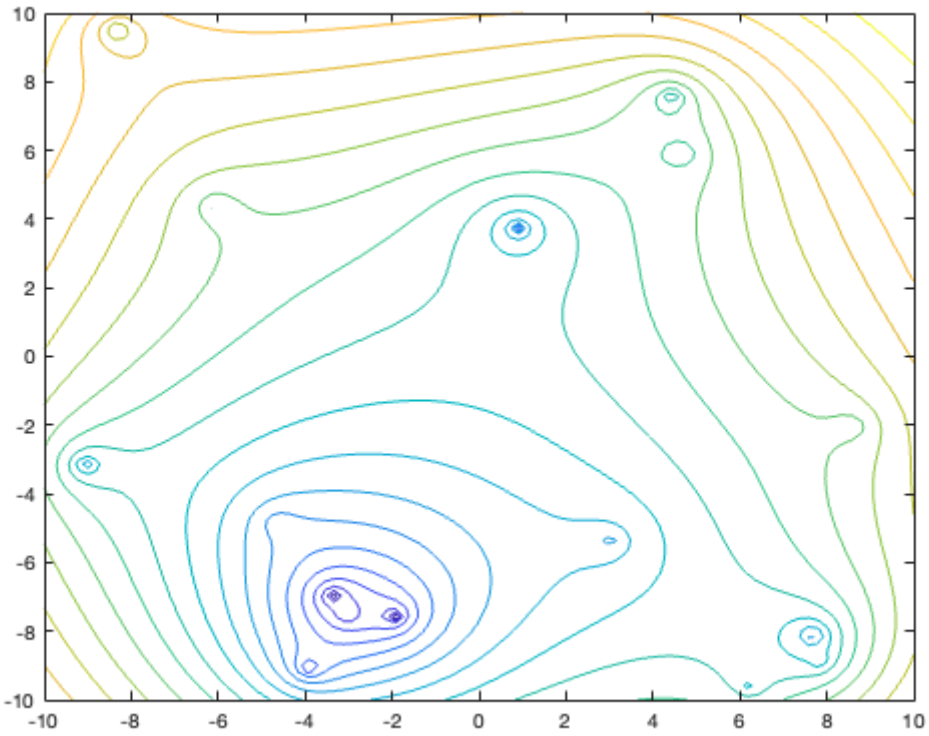
Example 29. 16 randomly placed v-slow vortices, half of which have the opposite orientation of the other half.

```
psi = 0;
for i=1:8
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed1,xg,yg);
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed2,xg,yg);
end

contour(xg,yg,psi,20)
```



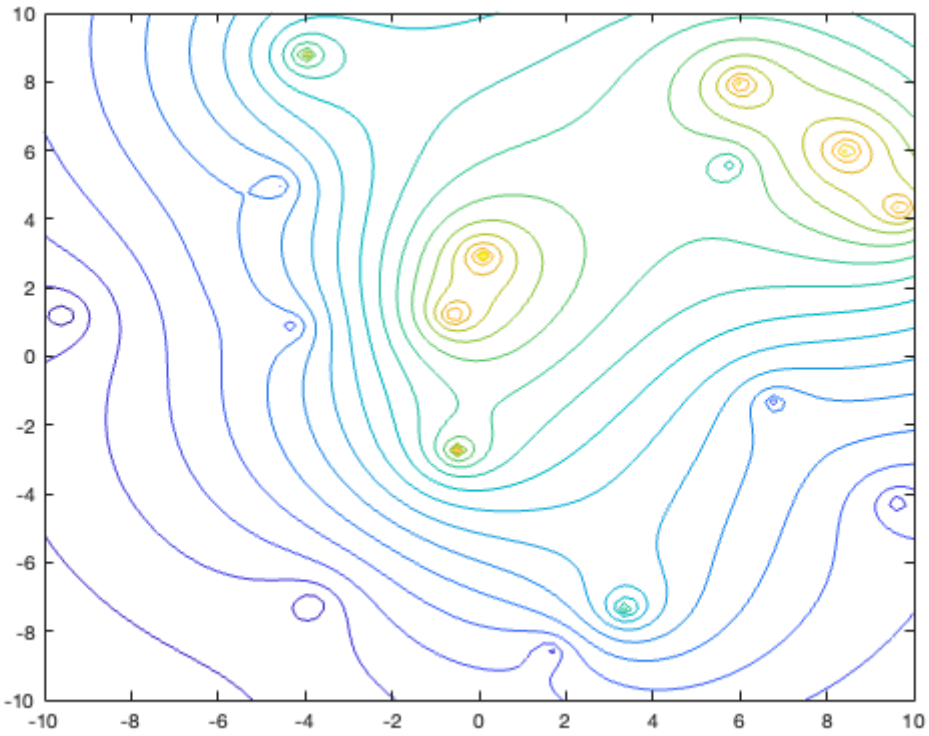
Example 30. 16 randomly placed v-slow vortices, half of which are stronger and half of which are weaker.

```
psi = 0;
for i=1:8
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed1,xg,yg);
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed3,xg,yg);
end

contour(xg,yg,psi,20)
```



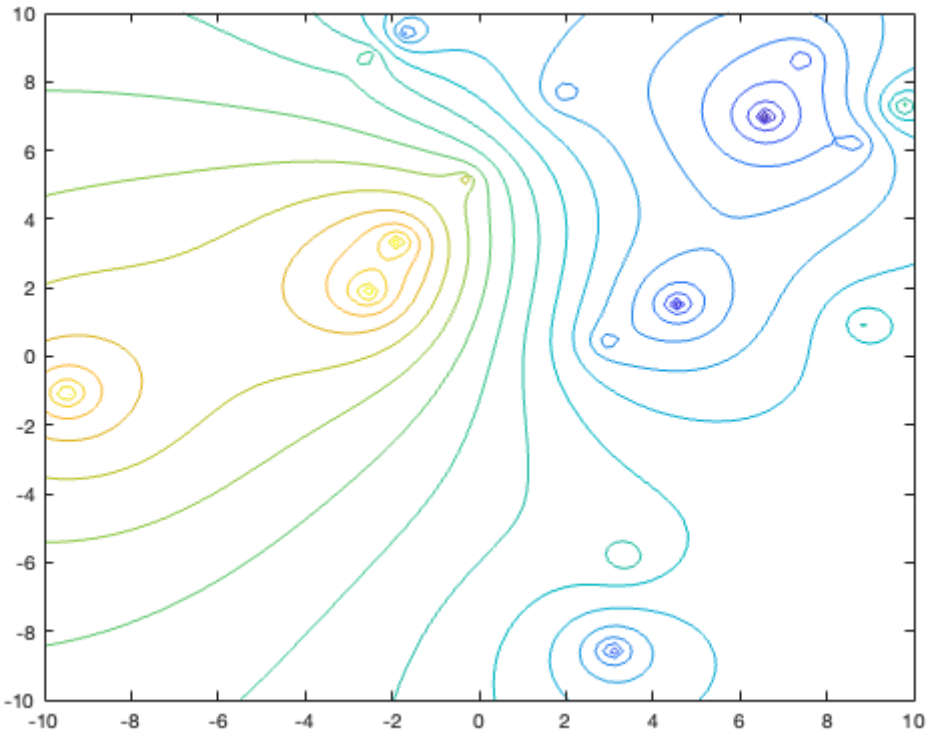
Example 31. 16 randomly placed v-slow vortices, half of which are stronger than the others and have the opposite orientation.

```
psi = 0;
for i=1:8
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed1,xg,yg);
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed4,xg,yg);
end

contour(xg,yg,psi,20)
```



Example 32. 16 randomly placed v-slow vortices, with one-fourth being counterclockwise and weak, one-fourth being counterclockwise and strong, one-fourth being clockwise and weak, and one-fourth being clockwise and strong.

```
psi = 0;
for i=1:4
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed1,xg,yg);
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed2,xg,yg);
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed3,xg,yg);
    psi = psi + vortex_psi(20*(rand-0.5),20*(rand-0.5),Speed4,xg,yg);
end

contour(xg,yg,psi,20)
```



Published with MATLAB® R2020b

B Appendix B: calc_gamma.m

```
function [Gamma, int_matrix] = calc_gamma(u_infty, semirad, circ, n)
conds = [];

% Calculation of angles
% "thetas" are the gridpoints of integration, starting at 0
% "theta0s" are the point vortex locations, shifted by dtheta/2
dtheta = 2*pi/n;
theta0s = linspace(0, 2*pi, n+1);
theta0s(end) = [];
thetas = theta0s;
theta0s = theta0s + dtheta/2;

% Calculation of coordinates
% "xs" and "ys" are gridpoints for integration
% "speeds" are the arclength per unit angle at each point
% "nxs" and "nys" are unit normal vecs at each point
% Analogous values for the theta0s are calculated
xs = cos(thetas);
ys = semirad*sin(thetas);
speeds = sqrt(semirad^2*cos(thetas).^2+sin(thetas).^2);
nxs = semirad*cos(thetas)./speeds;
nys = sin(thetas)./speeds;
speed0s = sqrt(semirad^2*cos(theta0s).^2+sin(theta0s).^2);
nx0s = semirad*cos(theta0s)./speed0s;
ny0s = sin(theta0s)./speed0s;
x0s = cos(theta0s);
y0s = semirad*sin(theta0s);

% Set up matrix system for integral equation
int_matrix = [];
for j=1:n
    theta0 = theta0s(j);
    x0 = x0s(j);
    y0 = y0s(j);
    denom = ((xs-x0).^2+(ys-y0).^2);
    numer = nx0s(j) .* (ys - y0) + ny0s(j) .* (-xs + x0);
    int_weight = (1/(2*pi*n))*numer./(denom.*speeds);
    int_matrix = [int_matrix; int_weight];
end

int_matrix = [int_matrix; ones(1, n)/n];
conds = [conds, cond(int_matrix)];
normal_vel = -u_infty*ny0s;
```

```
    % Add an extra equation for the circulation around the ellipse
    normal_vel = [normal_vel, circ];

    % Solve the system
    Gamma = int_matrix\normal_vel';

    % Checking that the normal velocity is actually zero
    for i=1:length(x0s)
        [u, v] = ellipse_velo(semirad, Gamma, x0s(i), y0s(i));
        vel = [u, v] + [0, u_infty];
        vel * [nx0s(i), ny0s(i)]';
    end
end
```

C Appendix C: gamma_grid.m

```
    function x = gamma_grid(semirad, Gamma, u_infty, m)
tol = 1e-10

xs = linspace(-2, 2, m);
ys = linspace(-2, 2, m);
Xp = [];
Yp = [];
Up = [];
Vp = [];

for i=1:m
    for j=1:m
        x = xs(i);
        y = ys(j);
        [u, v] = ellipse_velo(semirad, Gamma, x, y);
        if abs(u) < tol || x^2 + (y/semirad)^2 < 1
            u = 0;
        end
        if abs(v+u_infty) < tol || x^2 + (y/semirad)^2 < 1
            v = -u_infty;
        end
        if (x^2+y^2/semirad^2 > -1)
            Up = [Up,u];
            Vp = [Vp,v+u_infty];
            Xp = [Xp,x];
            Yp = [Yp,y];
        end
    end
end

figure
clf
quiver(Xp, Yp, Up, Vp)
[Xgr, Ygr] = meshgrid(xs, ys);
Ugr = reshape(Up, m, m);
Vgr = reshape(Vp, m, m);
streamslice(Xgr, Ygr, Ugr, Vgr)
title("Velocity field")
axis equal

end
```