The quantity f is proportional to the stream function of the flow, and $d^2f/d\eta^2$ is proportional to the shear. The similarity solution transforms the boundary layer equation into the ordinary nonlinear differential equation

$$2\frac{d^3f}{d\eta^3} + f\frac{d^2f}{d\eta^2} = 0 \tag{11.16}$$

where at $\eta = 0$

$$f = 0 \quad \frac{df}{d\eta} = 0 \tag{11.17a}$$

and as $\eta \to \infty$

$$\frac{df}{d\eta} \to 1 \tag{11.17b}$$

In order to solve Eq. (11.16), we reduce it to three first-order equations using the definitions

$$\begin{aligned} f_1 &= f \\ f_2 &= \frac{df}{d\eta} \\ f_3 &= \frac{d^2f}{d\eta^2} \end{aligned} \tag{11.18}$$

to obtain

$$\begin{aligned} \frac{df_1}{d\eta} &= f_2 \\ \frac{df_2}{d\eta} &= f_3 \\ \frac{df_3}{d\eta} &= -0.5 f_1 f_3 \end{aligned} \tag{11.19}$$

The boundary conditions at $\eta = 0$ become

$$f_1(0) = f_2(0) = 0 \tag{11.20a}$$

and as $\eta \to \infty$

$$f_2(\eta \to \infty) \to 1 \tag{11.20b}$$

To solve Eqs. (11.19) subject to the boundary conditions given by Eqs. (11.20), an iterative method is used. In this method, the values of $f_1(0)$ and $f_2(0)$ are given by Eq. (11.20a), and a guess for the wall shear stress $f_3(0)$ is assumed. Then the differential equations Eqs. (11.19) are integrated using `ode45` to find f_2 at the outer boundary at $\eta =$

η_{max}: $f_2(\eta_{max})$. For the correct value of $f_3(0)$, $f_2(\eta_{max}) \to 1.0$ when η_{max} is sufficiently large (see Eq. (11.20b)). In the script, the correct value of $f_3(0)$ is determined by iteration using `fzero`. The function *Blasius* is used by `ode45` to evaluate Eqs. (11.19). The function *Blasius2* is called by `fzero` to perform the iteration to find the value of $f_3(0)$ that produces $f_2(\eta_{max} \to \infty) \to 1.0$. This function also calls *Blasius*. These functions are given below.

The function *Blasius* is

```
function F = Blasius(x,y)
F = [y(2); y(3); -0.5*y(1)*y(3)];
```

and the function *Blasius2* is

```
function fn = Blasius2(fp0,EtaMax)
[eta ff] = ode45('Blasius', [0 EtaMax], [0 0 fp0]');
fn = 1.0-ff(end,2);
```

The script is

```
EtaMax = 20.0;
options = optimset('display','off');
shear0 = fzero('Blasius2', 0.3, options, EtaMax);
disp(['The shear stress at eta = 0 is: ' num2str(shear0)]);
[eta ff] = ode45('Blasius', [0 EtaMax], [0 0 shear0]');
plot(ff(:,1),eta,'k-',ff(:,2),eta,'k-.',ff(:,3),eta,'k--.')
axis([0 3 0 4])
ylabel('\eta')
xlabel('f, df/d\eta, d^2f/d\eta^2')
legend('f','df/d\eta','d^2f/d\eta^2',4)
```

When this script is executed we find that the shear stress is 0.33203 at $\eta = 0$, and we obtain the results shown in Figure 11.10.

11.3.3 Potential Flow

In incompressible potential flows, the velocity fields $\vec{u}$ are governed by

$$\nabla \cdot \vec{u} = 0$$

and

$$\nabla \times \vec{u} = 0$$

These conditions dictate that the velocity can be expressed as the gradient of a potential field, ϕ,

$$\vec{u} = \nabla\phi$$

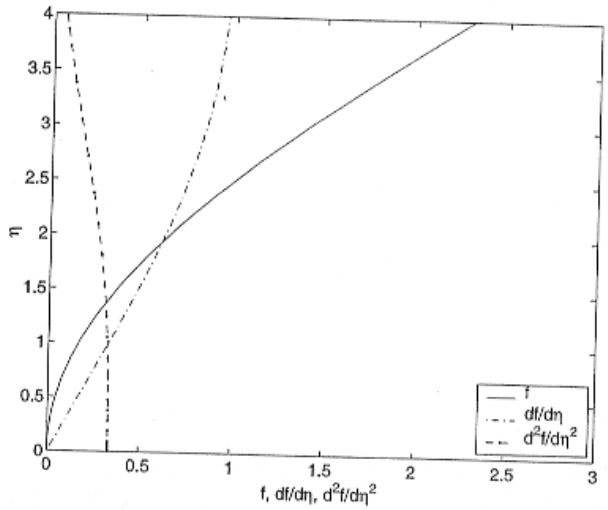

FIGURE 11.10.
Blasius boundary layer profiles of the stream function f_1, streamwise component of velocity f_2, and the shear f_3.

where ϕ satisfies Laplace's equation,

$$\nabla^2\phi = 0 \tag{11.21}$$

An alternative mathematical description for two-dimensional flows is obtained using the stream function ψ, where

$$u = \frac{\partial \psi}{\partial y}$$

$$v = -\frac{\partial \psi}{\partial x}$$

The stream function also satisfies Laplace's equation. Boundary conditions consist of the Neumann conditions, where the component of the velocity normal to a boundary is specified, or the Dirichlet conditions, where the value of ϕ is specified. At solid boundaries, the Neumann condition is $\vec{u} \cdot \hat{n} = 0$, where $\hat{n}$ is the unit normal to the boundary. In the following text, several methods for obtaining flow fields for two-dimensional potential flows are discussed. In two of these methods, the flows are constructed by adding together known potentials or stream functions. We now give four such quantities.

Sources and sinks

$$\phi_M = \frac{m}{2\pi}\ln r_M \qquad \psi_M = \frac{m}{2\pi}\theta_M$$

$$r_M^2 = (x - x_M)^2 + (y - y_M)^2 \qquad \theta_M = \tan^{-1}\frac{y - y_M}{x - x_M}$$

where (x_M,y_M) is the location of the source or sink and m is the source strength.

Doublets (dipoles)

$$\phi_K = \frac{K\cos\theta}{r_K} \qquad \psi_K = -\frac{K\sin\theta}{r_K}$$

$$r_K^2 = (x - x_K)^2 + (y - y_K)^2 \qquad \theta_K = \tan^{-1}\frac{y - y_K}{x - x_K}$$

where (x_K,y_K) is the location of the dipole and K is the dipole strength.

Vortices

$$\phi_\Gamma = \frac{\Gamma}{2\pi}\theta_\Gamma \qquad \psi_\Gamma = -\frac{\Gamma\ln r_\Gamma}{2\pi}$$

$$r_\Gamma^2 = (x - x_\Gamma)^2 + (y - y_\Gamma)^2 \qquad \theta_\Gamma = \tan^{-1}\frac{y - y_\Gamma}{x - x_\Gamma}$$

where (x_Γ,y_Γ) is the location of the vortex and Γ is the vortex strength;

Uniform flow field

$$\phi_U = Ux \qquad \psi_U = Uy$$

where U is the flow speed.

Thus, in general, one can form an additive combination of these different stream functions to simulate different flows around different shapes. Then, if ψ_s is the new streamline function,

$$\psi_s = \psi_M + \psi_K + \psi_\Gamma + \psi_U$$

Method 1: Determining the Streamline Pattern with `contour`. The first and easiest method of determining the streamline pattern of a flow is to plot the streamlines with `contour`. The following script plots the streamline ψ_s for a flow consisting of a uniform flow of speed U, a dipole of strength K located at (x_K,y_K), and a vortex of strength Γ located at (x_Γ,y_Γ). To illustrate this result, we choose the location of the dipole and vortex at (–1,–1) and give the strengths of each of these quantities the following values:

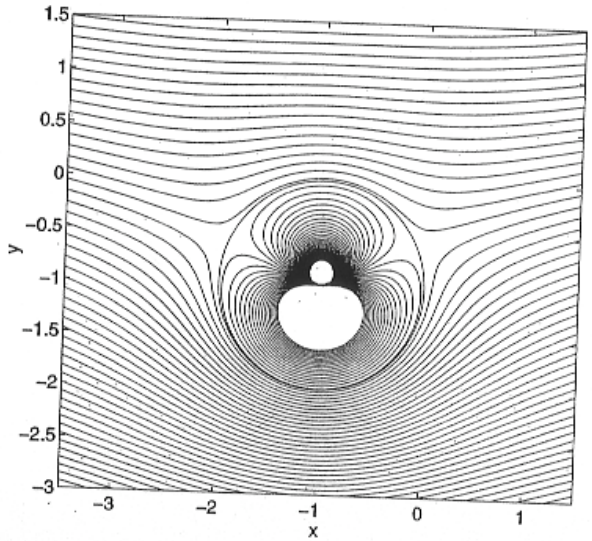

FIGURE 11.11
Streamlines for a cylinder with circulation in cross flow. Obtained from a contour plot of ψ_s.

$K = 5.0$ $(x_K, y_K) = (-1,-1)$
$\Gamma = 8\pi$ $(x_\Gamma, y_\Gamma) = (-1,-1)$
$U = 5.0$,

As illustrated in Figure 11.11, the resulting streamlines show flow about a cylinder with circulation. The main difficulty in obtaining these results is how to choose the contour levels to obtain a complete description of the flow. This can be accomplished using the value of ψ_s at the lower left corner of the domain for the minimum value and the value of ψ_s at the top middle of the domain for the maximum value. In Figure 11.11, the surface of a cylinder, which is also a streamline, has been superimposed on the streamlines.

```
nx =100; xmin = -3.5; xmax = 1.5;
ny = 100; ymin = -3.0; ymax = 1.5;
[x y] = meshgrid(linspace(xmin, xmax, nx),linspace(ymin, ymax, ny));
U0 = 5.0;
Gamma = 8*pi; xGamma = -1.0; yGamma = -1.0;
K = 5.0; xK = -1.0; yK = -1.0;
radius = inline('sqrt((x-x1).^2+(y-y1).^2)','x','y','x1','y1');
```



```
PsiK = K*sin(atan2(y-yK,x-xK))./radius(x,y,xK,yK);
PsiGamma = Gamma*log(radius(x,y,xGamma,yGamma))/2/pi;
StreamFunction = U0*y-PsiGamma -PsiK;
levmin = StreamFunction(1,nx);
levmax = StreamFunction(ny,nx/2);
levels = linspace(levmin,levmax,50)';
contour(x,y,StreamFunction,levels)
hold on
theta = linspace (0,2*pi);
plot(xGamma+cos(theta),yGamma+sin(theta),'k')
axis equal
axis([xmin xmax ymin ymax])
ylabel('y')
xlabel('x')
```

Method 2: Direct Calculation of Streamlines. A second method to obtain flow patterns is to use `fzero` to find specific streamlines. As an example we assume that the flow consists of a uniform stream of $U = 1$ in the positive y-direction, a source of strength $m = 4.0$ at $(0,-1)$, and a source of strength $m = -4.0$ at $(0,1)$. Thus,

$$\psi_{oval} = +\psi_U + \psi_{M_1} + \psi_{M_2}$$

These components produce a uniform flow over an oval-shaped body given by[5]

$$\frac{2x}{x^2 + y^2 - a^2} = \tan\frac{xU}{m/2\pi} \tag{11.22}$$

where U is the flow speed, m is the source strength, and a is a characteristic dimension.

A difficulty in using `fzero` in this example is the need to find a good starting guess. In the following script, this is done by finding the value of the stream function ψ at a set of x locations along $y = -2.0a$. Given this initial data, a streamline is computed by marching along it, starting at $y = -2.0a$. At each successive y location, `fzero` uses function *StreamFun* to determine the x location of the stream function; the value of x at the previous y location is used as the initial guess. A plot from the output of the script is given in Figure 11.12, where we have assumed that $a = 1$. The graph has been rotated 90° so that the flow is horizontal, which is traditional way of presenting it. The streamline that coincides with the boundary of the oval was not computed in this manner; it is plotted directly from Eq. (11.22). The script is

```
U = 1.0;a = 1.0;m = 4.0; co = m/(2*pi);
nPsi = 15; n = 30; yStart = -2.0*a;
xStart = linspace(0,2*a,nPsi);
y = linspace(-2*a,2*a,n);
x = zeros(1,n);
```

[5] L. M. Milne-Thomson, *Theoretical Hydrodynamics*, Dover, Mineola, NY, 1996, p. 216.

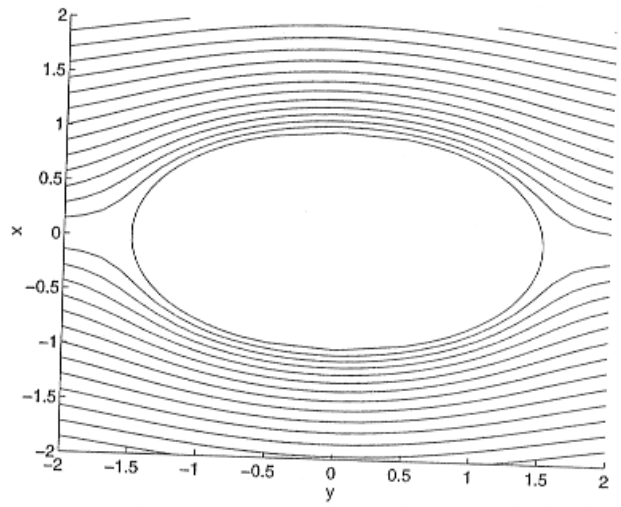

FIGURE 11.12
Streamlines for the oval given by Eq. (11.22).

```
StreamFun = inline('-U*x-co*(atan2(x,y+a)-atan2(x,y-a))- psi',...
                   'x','y','psi','U','co','a');
Psi = StreamFun(xStart,yStart,0,U,co,a);
options = optimset('display','off');
for j = 1:nPsi
 guess = xStart(j);
 for i = 1:n
  x(i) = fzero(StreamFun,guess,options,y(i),Psi(j),U,co,a);
  guess = x(i);
 end
 if j>1
  plot(y,x,'b',y,-x,'b')
 end
 hold on
end
axis([-2*a,2*a,-2*a,2*a])
ylabel('x')
xlabel('y')
xx = linspace(-1,1,40);
yy = sqrt(1-xx.^2+2*xx./tan(xx/co));
plot(yy,xx,'k',-yy,xx,'k')
```


where *nPsi* is the number of streamlines and *n* is the number of points computed along each streamline.

Method 3: Solving for the Flow Field with `pdetool`. A third method to obtain potential flow solutions is to use `pdetool` to solve Eq. (11.21) directly. Let us compute the flow field for a cylinder of diameter 1.0 that is placed in the center of a duct in which the flow speed is 10.0. In the `pdetool` window we select the axes of the window to extend from −3.5 to 3.5 in *x*-direction and from −2.5 to 2.5 in *y*-direction. We then create a circle (*C*1) of radius 1.0 centered at (0,0) and a rectangle (*R*1) of dimensions 6.0 wide by 5.0 high centered at (0,0). We alter the *Set Formula* to *R*1−*C*1 and then select the *Boundary Mode*.

We then select *Specify Boundary Conditions*, and on the circle and the top and bottom boundaries of the rectangle we select the Neumann boundary condition $\hat{n}\cdot\nabla\phi = 0$. To achieve this specification we set $c = 1$, $q = 0$, and $g = 0$. On the left side of the rectangle we select the Neumann boundary condition $\hat{n}\cdot\nabla\phi = 10$—that is, we set $c = 1$, $q = 0$, and $g = 10$. On the right side we set $\hat{n}\cdot\nabla\phi = -10$—that is, we set $c = 1$, $q = 0$, and $g = -10$. This makes the mean flow go from left to right, since the unit normal $\hat{n}$ to the boundary in `pdetool` is directed from the boundary toward the flow domain.

Next we initialize the mesh and refine it twice. The partial differential equation is specified by selecting *elliptic* from *PDE Specification* in the *PDE* pull-down menu. Our equation is $\nabla^2 u = 0$, where in the present case the MATLAB variable u represents ϕ; therefore, we set $c = 1$, $a = 0$, and $f = 0$. We solve for u and produce a vector plot of the velocity ∇u on top of a contour plot of u as shown in Figure 11.13.

To obtain more detailed results, we export the solution u and the mesh coordinates p, e, and t to the MATLAB command window. A plot of the velocity along any vertical or horizontal line can be obtained in the following manner. Consider the horizontal velocity along the vertical line $x = 0$ extending from the top of the cylinder to the top of the rectangle. To obtain a plot of this velocity distribution, the script given below is used. In the script we first create a rectangular grid in the area of interest, say $-0.5 \le x \le 0.5$ ($n_x = 9$ points) and $0.5 \le y \le 2.5$ ($n_y = 25$ points) with `tri2grid`. We then use `gradient` to obtain the difference field. Finally, the horizontal component of the gradient is obtained along the line $x = 0$ by dividing the appropriate differences by the grid spacing in the *x*-direction. The resulting array of velocities is called *ux* in the script. A plot of *ux* versus *y* is given in Figure 11.14. For a cylinder in an infinite flow field, the maximum velocity, which occurs on the top and bottom of the cylinder (±90° from the flow direction), is $2U$. In the present case, the cylinder is in a duct created by the top and bottom of the rectangle. The maximum velocity is again at the 90° position on the cylinder, and the value is $ux(1) = 2.264U$.

```
nx = 9; xmin = -0.5; xmax = 0.5;
x = linspace(xmin, xmax, nx);
ny = 25; ymin = 0.5; ymax = 2.5;
y = linspace(ymin, ymax, ny);
uxy = tri2grid(p,t,u,x,y);
[DX,DY] = gradient(uxy);
ux = -DX(:,(nx-1)/2)/((xmax-xmin)/(nx-1));
plot(ux,y)
```

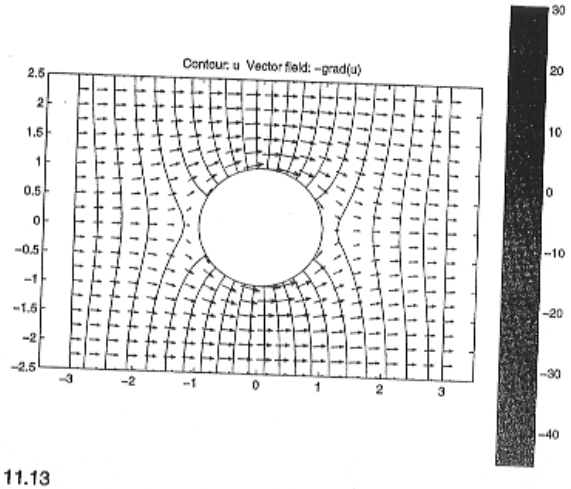

FIGURE 11.13
Velocity potential (contours) and velocity vectors from the direct solution of Eq. (11.21).

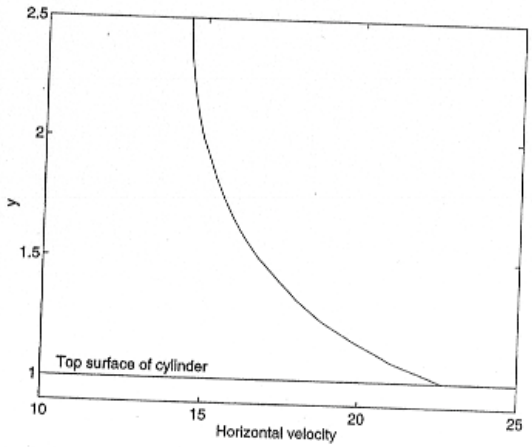

FIGURE 11.14
Horizontal velocity distribution along a vertical line extending from the top surface of a cylinder.


```
axis([10,25,0.9,2.5])
ylabel('y')
xlabel('Horizontal velocity')
hold on
plot([10,25],[1,1])
text(10.5,1.05,'Top surface of cylinder')
ux1 = max(ux)
```

EXERCISES

11.1 Obtain the flow fields in two ducts with the same cross-sectional area, but with one whose cross-sectional shape is square and one whose shape is rectangular. The rectangular shaped cross section has the length of one side four times that of the other. Compare the volume flow rates of the two ducts using `pdetool`. The governing equation is that given by Eq. (11.7). Assume that the right-hand side of Eq. (11.7) is the same for each duct, say 1.0—that is, they have the same fluid and pressure gradient. Also, the boundary condition at the duct wall is $u = 0$. Export each solution to the MATLAB command window and use the procedure illustrated in Section 11.2.1 to obtain

$$Q_{sq}/Q_{rect} = 2$$

where Q is the flow rate.

11.2 The flow about a thin symmetric airfoil can be approximated by potential flow theory.[6] The chord of the airfoil extends along the x axis from $x = 0$ to $x = c$, and is represented by a vortex sheet whose strength $\gamma(x)$ is given by

$$\gamma(\theta) = 2\alpha V_\infty \frac{1+\cos\theta}{\sin\theta}$$

where

$$x = \frac{c}{2}(1-\cos\theta) \qquad 0 \le \theta \le \pi$$

α is the angle of attack (in radians) of the incoming flow relative to the x axis, and V_∞ is the flow speed. Consider the vortex sheet to be approximated by a set of N discrete vortices separated by a distance $\Delta x = c/N$ with strength $\Gamma_i = \gamma(\theta_i)\Delta x$. Using Method 1 of Section 11.3.3, draw the streamlines of this flow for $\alpha = 10°$, $c = 2$ m, and $V_\infty = 100$ m/s. The results should look like those shown in Figure 11.15.

11.3 Consider the flow field around a cylinder in a duct as shown in Figure 11.16. Assume potential flow and use `pdetool` to compute the streamlines. Export the mesh and solution variables to the MATLAB command window and compute the velocity distribution along the bottom wall of the channel. The results should look like those presented in Figure 11.17.

11.4 Consider a potential flow over a cylinder that is placed near a wall as shown in Figure 11.18. Represent the flow over the cylinder and wall with a uniform flow of speed $U = 1.0$ m/s and two dipoles located at $(x,y) = (0,0.25D)$ and $(0,-0.25D)$. The velocity potential for each dipole is

$$\phi = \frac{UD^2}{4r}\cos\theta$$

[6] See, for example, J. D. Anderson, *Fundamentals of Aerodynamics*, McGraw-Hill, New York, 1991, Chapter 4.