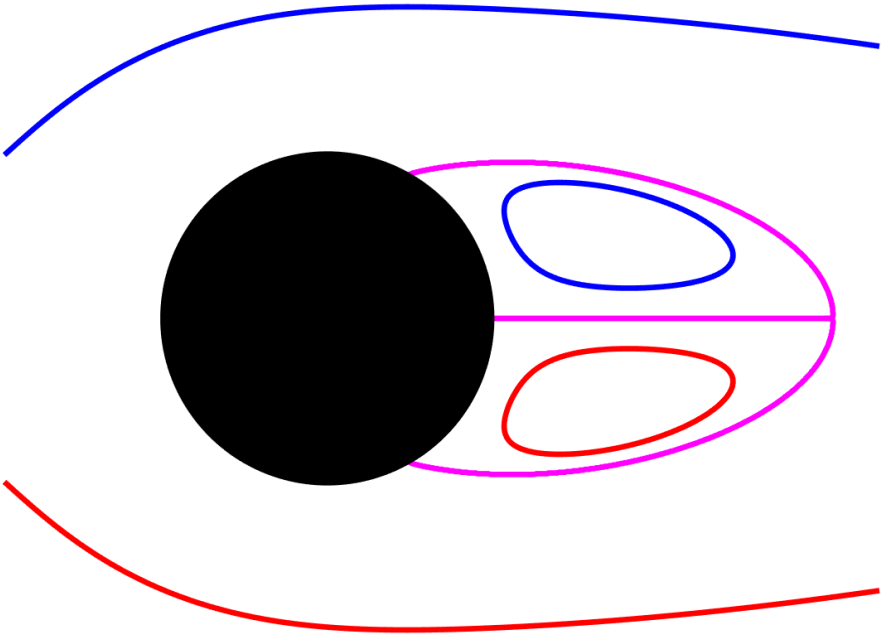
Flow Around a Cylinder

Lecture Notes for coursera

Jeffrey R. Chasnov



THE HONG KONG UNIVERSITY OF
SCIENCE AND TECHNOLOGY

The Hong Kong University of Science and Technology
Department of Mathematics
Clear Water Bay, Kowloon
Hong Kong

Preface

View the Deep Dive video **Flow Around a Cylinder** on YouTube

These are my lecture notes for Mathematics for Engineers: The Capstone Course. Students will learn how to solve for the steady and unsteady two-dimensional flow around an infinite cylinder. Before students take this course, they should have some basic knowledge of single-variable calculus, vector calculus, differential equations, matrix algebra and numerical methods. Students should also be able to program in MATLAB.

I have divided these notes into chapters called Lectures, with each Lecture corresponding to a video on Coursera. I have also uploaded all my Coursera videos to YouTube, and links are placed at the top of each Lecture.

The course is divided into three weeks. The first week introduces the Navier-Stokes equations and defines the flow geometry. The second week details the numerical methods we use to compute steady flow solutions. And the third week details the methods we use to compute unsteady flow solutions and the Kármán vortex street.

After every Lecture of Week One, there are problems to solve. There are problems in Week Two and Week Three also, but these weeks culminate in a substantial MATLAB program to write. Solutions to the problems and Learner Templates for the MATLAB programs can be found in the Appendix.

JEFFREY R. CHASNOV
Hong Kong
Feb 2022

Contents

Week I

Governing Equations

In this week's lectures, we introduce the Navier-Stokes equations and the flow around an infinite circular cylinder. Our flow field will be two dimensional and we write the Navier-Stokes equations using a scalar vorticity and stream function. To take advantage of the circular symmetry, we formulate the equations in polar coordinates. A further simplification of the Laplacian operator is made by defining log-polar coordinates.

Lecture 1 | Navier-Stokes equations

View this lecture on YouTube

For an incompressible fluid, the continuity and Navier-Stokes equations are written in vector notation as

$$\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0, \qquad \frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u} = -\frac{1}{\rho}\boldsymbol{\nabla} p + \nu \nabla^2 \boldsymbol{u}.$$

The first equation states that the velocity field is a divergence-free (or solenoidal) vector field. It is an expression of the incompressibility condition—that the density of the fluid is everywhere constant. The second equation expresses Newton's law $F = ma$ applied to a fluid, and its derivation is usually presented in a first course on Fluid Mechanics.

Some students may have trouble deciphering the compact vector notation. To illustrate how to unpack these equations, we use Cartesian coordinates and write

$$\boldsymbol{u} = u(x, y, z)\boldsymbol{i} + v(x, y, z)\boldsymbol{j} + w(x, y, z)\boldsymbol{k},$$

where the three components also depend on the time t. The continuity equation expands to

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0;$$

and the Navier-Stokes equation expands into the three-component equations

$$\begin{aligned}
\frac{\partial u}{\partial t} + \left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} \right) &= -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right), \\
\frac{\partial v}{\partial t} + \left(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} \right) &= -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2} \right), \\
\frac{\partial w}{\partial t} + \left(u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} \right) &= -\frac{1}{\rho}\frac{\partial p}{\partial z} + \nu \left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} \right).
\end{aligned}$$

The goal of our capstone course is to solve numerically a two-dimensional version of these equations for the flow around a cylinder.

Problems for Lecture 1

1. Plane Couette flow consists of an incompressible fluid flowing between two infinite plates separated by a distance d. The lower plate is stationary, and the upper plate is moving to the right with velocity U. The pressure p is constant. Choose a coordinate system so that the flow is in the x-direction and the y-axis is perpendicular to the two plates. Find the solution for the velocity field of the form $\boldsymbol{u}(x,y,z) = u(y)\boldsymbol{i}$.

2. Channel flow, or Poiseuille flow, consists of an incompressible fluid flowing between two infinite plates separated by a distance d, but with both plates stationary. Here, the fluid flows between the two plates in the direction of a constant pressure gradient. Choose a coordinate system so that the fluid flow is in the x-direction and the y-axis is perpendicular to the two plates. Find the solution for the velocity field of the form $\boldsymbol{u}(x,y,z) = u(y)\boldsymbol{i}$, with $p = p(x)$, $dp/dx = -G$ and G is a positive constant.

3. Pipe flow consists of an incompressible fluid flowing through a pipe of circular cross-section radius R, with a constant pressure gradient along the pipe length. Choose a coordinate system so that the pressure gradient is in the x-direction and the y- and z-axes are perpendicular to the sides of the pipe. Find the solution for the velocity field of the form $\boldsymbol{u} = u(y,z)\boldsymbol{i}$, with $p = p(x)$, $dp/dx = -G$ and G is a positive constant. Use polar coordinates.

Solutions to the Problems

Lecture 2 | Vorticity equation

View this lecture on YouTube

The equation for the vorticity, $\boldsymbol{\omega} = \boldsymbol{\nabla} \times \boldsymbol{u}$, may be found by taking the curl of the Navier-Stokes equation; that is

$$\boldsymbol{\nabla} \times \left\{ \frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u} \right\} = \boldsymbol{\nabla} \times \left\{ -\frac{1}{\rho}\boldsymbol{\nabla} p + \nu \nabla^2 \boldsymbol{u} \right\}.$$

Examining each term, we have

$$\boldsymbol{\nabla} \times \left\{ \frac{\partial \boldsymbol{u}}{\partial t} \right\} = \frac{\partial}{\partial t}\left(\boldsymbol{\nabla} \times \boldsymbol{u}\right) = \frac{\partial \boldsymbol{\omega}}{\partial t} \qquad \text{and} \qquad \boldsymbol{\nabla} \times \left\{ \nu \nabla^2 \boldsymbol{u} \right\} = \nu \nabla^2 \left(\boldsymbol{\nabla} \times \boldsymbol{u}\right) = \nu \nabla^2 \boldsymbol{\omega}.$$

And because the curl of a gradient is zero,

$$\boldsymbol{\nabla} \times \{ -\boldsymbol{\nabla} p / \rho \} = 0.$$

To simplify the curl of the convection term, one writes

$$\boldsymbol{\nabla} \times \{ (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u} \} = \boldsymbol{\nabla} \times \left\{ \frac{1}{2}\boldsymbol{\nabla}(\boldsymbol{u}^2) + \boldsymbol{\omega} \times \boldsymbol{u} \right\} = \boldsymbol{\nabla} \times \left(\boldsymbol{\omega} \times \boldsymbol{u} \right);$$

and the vorticity equation becomes

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \boldsymbol{\nabla} \times \left(\boldsymbol{\omega} \times \boldsymbol{u} \right) = \nu \nabla^2 \boldsymbol{\omega}.$$

An alternative form expands the convection term to obtain

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{\omega} = (\boldsymbol{\omega} \cdot \boldsymbol{\nabla})\boldsymbol{u} + \nu \nabla^2 \boldsymbol{\omega}.$$

Compared to the Navier-Stokes equation for $\boldsymbol{u}$, there is an extra term on the right-hand side of this equation, $(\boldsymbol{\omega} \cdot \boldsymbol{\nabla})\boldsymbol{u}$. This term is called the vortex stretching term and is often used to explain the energy cascade in three-dimensional turbulence.

Problems for Lecture 2

1. Prove the following equalities:

a) $\boldsymbol{\nabla} \times \{(\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u}\} = \boldsymbol{\nabla} \times (\boldsymbol{\omega} \times \boldsymbol{u})$;

b) $\boldsymbol{\nabla} \times (\boldsymbol{\omega} \times \boldsymbol{u}) = (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{\omega} - (\boldsymbol{\omega} \cdot \boldsymbol{\nabla})\boldsymbol{u}$.

You can use the facts that the curl of a gradient and the divergence of a curl are equal to zero, and the general vector identity

$$\boldsymbol{\nabla} \times (\boldsymbol{u} \times \boldsymbol{v}) = \boldsymbol{u}(\boldsymbol{\nabla} \cdot \boldsymbol{v}) - \boldsymbol{v}(\boldsymbol{\nabla} \cdot \boldsymbol{u}) + (\boldsymbol{v} \cdot \boldsymbol{\nabla})\boldsymbol{u} - (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{v}.$$

You will also need to use $\boldsymbol{\omega} = \boldsymbol{\nabla} \times \boldsymbol{u}$ and $\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0$. However, you will need to prove that

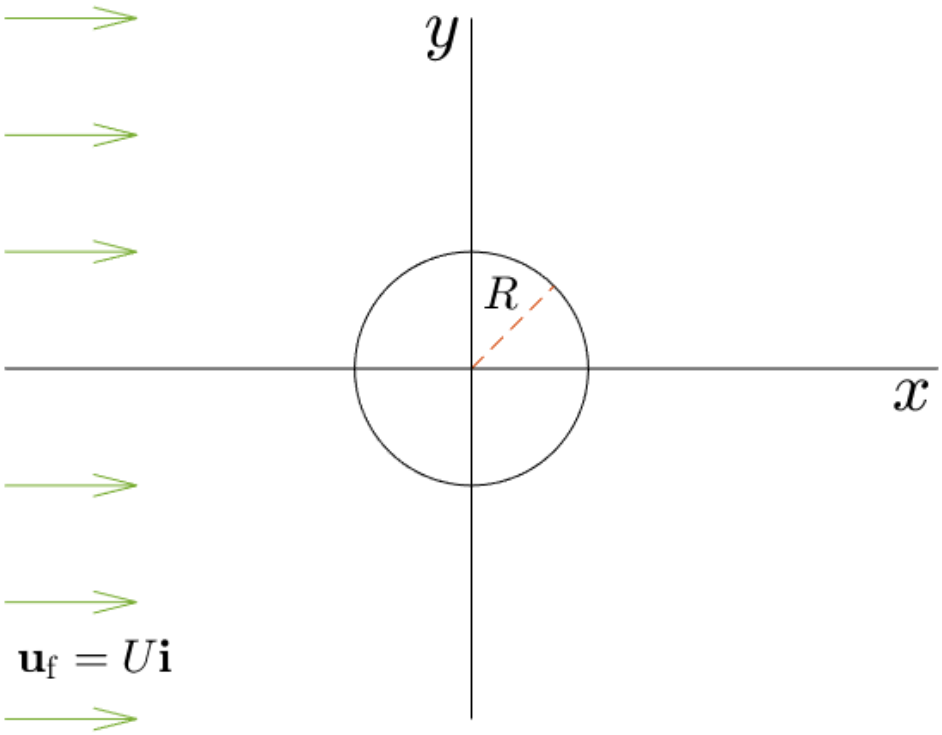
$$\boldsymbol{u} \times (\boldsymbol{\nabla} \times \boldsymbol{u}) = \frac{1}{2}\boldsymbol{\nabla}(\boldsymbol{u} \cdot \boldsymbol{u}) - (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u}.$$

Solutions to the Problems

Lecture 3 | Geometry of the flow

View this lecture on YouTube

We consider flow around an infinite cylinder with cross-sectional radius R. We define our x-y coordinates to lie in the cross-sectional plane, and orient the axes so that the free-stream velocity, $\boldsymbol{u}_f = U\boldsymbol{i}$, is in the x-direction. The geometry of the flow in the plane is shown in the figure below.



For the nonturbulent flows considered here, we assume that the velocity field vectors lie in the x-y plane and have no z-component, and that the flow field is independent of the z coordinate. And for steady flows only, we assume that the flow field in the bottom half of the plane ($y < 0$) is a mirror image of the flow field in the top half of the plane ($y > 0$). That is, with velocity field given by $\boldsymbol{u} = u(x,y)\boldsymbol{i} + v(x,y)\boldsymbol{j}$, we assume $u(x,-y) = u(x,y)$ and $v(x,-y) = -v(x,y)$.

Problems for Lecture 3

1. In the cross-sectional plane, express the cylinder boundary in Cartesian and polar coordinates. Express the free-stream velocity in polar coordinates.

Solutions to the Problems

Lecture 4 | Two-dimensional flow

View this lecture on YouTube

The vorticity equation is given by

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{\omega} = (\boldsymbol{\omega} \cdot \boldsymbol{\nabla})\mathbf{u} + \nu \nabla^2 \boldsymbol{\omega},$$

where $\boldsymbol{\omega} = \boldsymbol{\nabla} \times \boldsymbol{u}$. For nonturbulent flow past an infinite cylinder, one expects that the velocity field will lie in the cross-sectional plane of the cylinder and be independent of the z coordinate.

To best model the boundary conditions on the cylinder, we will use polar coordinates and write

$$\boldsymbol{u} = u_r(r, \theta)\hat{r} + u_\theta(r, \theta)\hat{\theta},$$

where the components of the velocity may also depend on time.

In polar coordinates, the vorticity field becomes

$$\boldsymbol{\omega} = \boldsymbol{\nabla} \times \boldsymbol{u} = \frac{1}{r}\left(\frac{\partial}{\partial r}(r u_\theta) - \frac{\partial u_r}{\partial \theta}\right)\boldsymbol{k},$$

and has only a single component in the z-axis direction. We can then define a scalar vorticity field by

$$\omega = \frac{1}{r}\left(\frac{\partial}{\partial r}(r u_\theta) - \frac{\partial u_r}{\partial \theta}\right).$$

And since

$$(\boldsymbol{\omega} \cdot \boldsymbol{\nabla})\boldsymbol{u} = \omega \frac{\partial \boldsymbol{u}}{\partial z} = 0,$$

and

$$\boldsymbol{\nabla} = \hat{r}\frac{\partial}{\partial r} + \hat{\theta}\frac{1}{r}\frac{\partial}{\partial \theta}, \qquad \nabla^2 = \frac{1}{r^2}\left(\left(r\frac{\partial}{\partial r}\right)\left(r\frac{\partial}{\partial r}\right) + \frac{\partial^2}{\partial \theta^2}\right),$$

the vorticity vector equation in polar coordinates becomes the scalar equation

$$\frac{\partial \omega}{\partial t} + u_r\frac{\partial \omega}{\partial r} + u_\theta\frac{1}{r}\frac{\partial \omega}{\partial \theta} = \frac{\nu}{r^2}\left(\left(r\frac{\partial}{\partial r}\right)\left(r\frac{\partial \omega}{\partial r}\right) + \frac{\partial^2 \omega}{\partial \theta^2}\right).$$

Problems for Lecture 4

1. With

$$\boldsymbol{\omega} = \omega(x,y)\boldsymbol{k}, \qquad \boldsymbol{u} = u(x,y)\boldsymbol{i} + v(x,y)\boldsymbol{j},$$

show by direct calculation that

$$\boldsymbol{\nabla} \times (\boldsymbol{\omega} \times \boldsymbol{u}) = \left(u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y} \right) \boldsymbol{k}.$$

2. Define the scalar vorticity in Cartesian coordinates.

3. Write the scalar vorticity equation in Cartesian coordinates.

Solutions to the Problems

Lecture 5 | Stream function

View this lecture on YouTube

The incompressibility condition in polar coordinates is given by

$$\boldsymbol{\nabla} \cdot \boldsymbol{u} = \frac{1}{r}\left(\frac{\partial}{\partial r}\left(r u_r\right) + \frac{\partial u_\theta}{\partial \theta}\right) = 0.$$

To automatically satisfy this condition, we make use of the equality of mixed partials and define a stream function ψ by

$$r u_r = \frac{\partial \psi}{\partial \theta}, \qquad u_\theta = -\frac{\partial \psi}{\partial r}.$$

The definition of the scalar vorticity then becomes

$$\begin{aligned}\omega &= \frac{1}{r}\left(\frac{\partial}{\partial r}(r u_\theta) - \frac{\partial u_r}{\partial \theta}\right) \\ &= -\frac{1}{r^2}\left(r\frac{\partial}{\partial r}\left(r\frac{\partial \psi}{\partial r}\right) + \frac{\partial^2 \psi}{\partial \theta^2}\right),\end{aligned}$$

which is simply the equation

$$\nabla^2 \psi = -\omega$$

in polar coordinates.

Problems for Lecture 5

1. Define the stream function in Cartesian coordinates.

2. Using Cartesian coordinates, write the scalar vorticity in terms of the stream function.

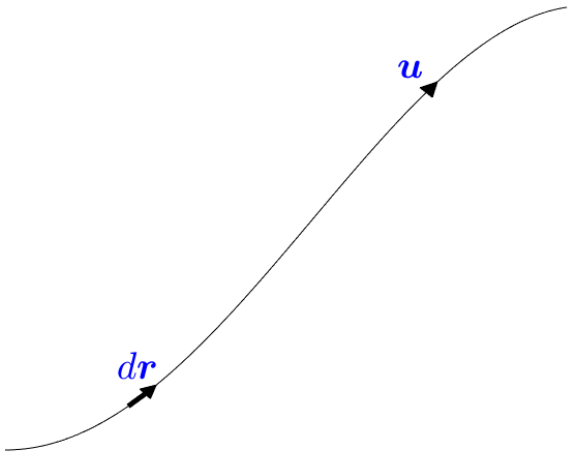
3. In general, a solenoidal vector field $\boldsymbol{u}$ that satisfies $\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0$ admits a vector potential $\boldsymbol{A}$ such that $\boldsymbol{u} = \boldsymbol{\nabla} \times \boldsymbol{A}$. Using Cartesian coordinates in two dimensions, show that the stream function can be identified as the third component of a vector potential $\boldsymbol{A}$.

Solutions to the Problems

Lecture 6 | Streamlines

View this lecture on YouTube

The streamlines in a steady flow show the direction of fluid motion. A streamline is everywhere tangent to the velocity field of the flow (see figure).



With infinitesimal vector displacement $d\boldsymbol{r} = dr\hat{r} + rd\theta\hat{\theta}$ along a streamline, and with the velocity field parallel to $d\boldsymbol{r}$ so that their cross-product is zero, we have

$$0 = \boldsymbol{u} \times d\boldsymbol{r} = (u_r\hat{r} + u_\theta\hat{\theta}) \times (dr\hat{r} + rd\theta\hat{\theta}) = (-u_\theta dr + ru_r d\theta))\boldsymbol{k}.$$

But since the stream function ψ is defined by

$$ru_r = \frac{\partial\psi}{\partial\theta}, \qquad u_\theta = -\frac{\partial\psi}{\partial r},$$

we have determined that

$$0 = \left(\frac{\partial\psi}{\partial r}dr + \frac{\partial\psi}{\partial\theta}d\theta\right)\boldsymbol{k} = d\psi\boldsymbol{k},$$

or $d\psi = 0$ along streamlines. The streamlines are therefore curves of constant ψ, and a plot of the contour lines of the stream function thus provides us an excellent visual representation of a steady two-dimensional flow.

Problems for Lecture 6

1. Use Cartesian coordinates to show that the stream function is constant along streamlines.

Solutions to the Problems

Lecture 7 | Reynolds number

View this lecture on YouTube

The governing equations for a two-dimensional flow in terms of the stream function and scalar vorticity field are given by

$$\nabla^2\psi = -\omega, \qquad \frac{\partial\omega}{\partial t} + \frac{1}{r}\left(\frac{\partial\psi}{\partial\theta}\frac{\partial\omega}{\partial r} - \frac{\partial\psi}{\partial r}\frac{\partial\omega}{\partial\theta}\right) = \nu\nabla^2\omega,$$

where ∇^2 is the two-dimensional Laplacian in polar coordinates.

We now consider flow past an infinite circular cylinder of radius R, with free-stream velocity $\boldsymbol{u} = U\boldsymbol{i}$. The units of our problem are time t and length l, and $[R] = l$, $[U] = lt^{-1}$, and the various terms in the equations have units

$$\left[\frac{\partial}{\partial t}\right] = t^{-1}, \quad \left[\frac{\partial}{\partial r}\right] = l^{-1}, \quad [\nabla^2] = l^{-2}, \quad [\nu] = l^2t^{-1}$$
$$[\psi] = l^2t^{-1}, \quad [\omega] = t^{-1}.$$

If we nondimensionalize using R and U, the net result is the replacement of the viscosity ν by a dimensionless grouping of parameters, that is,

$$\nu \to \frac{\nu}{UR}.$$

It is standard here to define the Reynolds number Re in terms of the diameter of the cylinder instead of the radius, so that with

$$\mathrm{Re} = \frac{2UR}{\nu},$$

the nondimensional governing equations (with all variables now dimensionless) become

$$\nabla^2\psi = -\omega, \qquad \frac{\partial\omega}{\partial t} + \frac{1}{r}\left(\frac{\partial\psi}{\partial\theta}\frac{\partial\omega}{\partial r} - \frac{\partial\psi}{\partial r}\frac{\partial\omega}{\partial\theta}\right) = \frac{2}{\mathrm{Re}}\nabla^2\omega.$$

Problems for Lecture 7

1. From the nondimensional equations for the stream function and the scalar vorticity, determine a single equation for the stream function when $\mathrm{Re} = 0$.

Solutions to the Problems

Lecture 8 | Log-polar coordinates

View this lecture on YouTube

The recurring factor $r\partial/\partial r$ in the polar coordinate Laplacian is awkward to discretize and we look for a change of variables $r = r(\xi)$, where

$$r\frac{\partial}{\partial r} = \frac{\partial}{\partial \xi}.$$

Now,

$$\frac{\partial}{\partial \xi} = \frac{dr}{d\xi}\frac{\partial}{\partial r},$$

so that we require

$$\frac{dr}{d\xi} = r.$$

This simple differential equation can be solved if we take as our boundary condition $\xi = 0$ when $r = 1$, corresponding to points lying on the boundary of the circular cylinder. The solution of the differential equation is therefore given by

$$r = e^{\xi}.$$

The Laplacian in these log-polar coordinates then becomes

$$\begin{aligned}\nabla^2 &= \frac{1}{r^2}\left(\left(r\frac{\partial}{\partial r}\right)\left(r\frac{\partial}{\partial r}\right) + \frac{\partial^2}{\partial \theta^2}\right)\\ &= e^{-2\xi}\left(\frac{\partial^2}{\partial \xi^2} + \frac{\partial^2}{\partial \theta^2}\right);\end{aligned}$$

and the governing equations become

$$\begin{aligned}\left(\frac{\partial^2 \psi}{\partial \xi^2} + \frac{\partial^2 \psi}{\partial \theta^2}\right) &= -e^{2\xi}\omega,\\ \frac{\partial \omega}{\partial t} + e^{-2\xi}\left(\frac{\partial \psi}{\partial \theta}\frac{\partial \omega}{\partial \xi} - \frac{\partial \psi}{\partial \xi}\frac{\partial \omega}{\partial \theta}\right) &= \frac{2}{\mathrm{Re}}e^{-2\xi}\left(\frac{\partial^2 \omega}{\partial \xi^2} + \frac{\partial^2 \omega}{\partial \theta^2}\right).\end{aligned}$$

Problems for Lecture 8

1. Determine the governing equations for a steady flow in log-polar coordinates. Try to write the equations as neatly as possible.

Solutions to the Problems

Week II

Steady Flows

In this week's lectures, we show how to solve the stream function and scalar vorticity equations to obtain steady solutions. These steady solutions for the fields do not depend on time. We discretize the equations using a second-order finite difference method and the SOR iteration method. Low Reynolds number iterations can be initialized with zero flow fields except for the free-stream condition on the stream function. We derive a key boundary condition for the scalar vorticity on the cylinder.

For the computational project, students will compute and plot the stream function for $Re = 10$*. Experiments have shown that the steady solutions are stable for* $Re < 46$*. The steady solutions still exist for higher Reynolds numbers, but they are unstable and not observed in Nature.*

Lecture 9 | Finite difference method

View this lecture on YouTube

A finite difference approximation requires a grid in (ξ, θ) space. With velocity field $\boldsymbol{u} = u(x,y)\boldsymbol{i} + v(x,y)\boldsymbol{j}$, we assume that u is even in y and v is odd; that is,

$$u(x,-y) = u(x,y), \qquad v(x,-y) = -v(x,y).$$

If a function is even in a variable, then its partial derivative in that variable is odd, and vice-a-versa. Since $u = \partial\psi/\partial y$ and $\omega = \partial v/\partial x - \partial u/\partial y$, both the stream function and scalar vorticity must be odd in y:

$$\psi(x,-y) = -\psi(x,y), \qquad \omega(x,-y) = -\omega(x,y).$$

We need then define a grid only for the upper half of the x-y plane, and take as our boundary conditions at $y = 0$ the values $\psi = \omega = 0$.

In log-polar coordinates, we can then define our grid as $0 \le \xi \le \xi_{\max}$ and $0 \le \theta \le \pi$, and our computational domain forms a rectangle without any holes. The sides of this rectangle correspond to the cylinder boundary ($\xi = 0$), the free stream ($\xi = \xi_{\max}$), the midline behind the cylinder ($\theta = 0$), and the midline in front of the cylinder ($\theta = \pi$).

We discretize the computational domain using square grid cells, and write

$$\xi_i = (i-1)h, \quad i = 1,2,\ldots,n; \qquad \theta_j = (j-1)h, \quad j = 1,2,\ldots,m,$$

where n and m are the number of grid points (including boundary points) in the ξ and θ directions, and h is the side length of a grid cell. Because $0 \le \theta \le \pi$, the value of h must satisfy $h = \pi/(m-1)$, and the maximum value of ξ is given by $\xi_{\max} = (n-1)\pi/(m-1)$. The radius of the computational domain is therefore given by $e^{\xi_{\max}}$, which is to be compared to a cylinder radius of one. The choice $n = m$ yields $e^{\xi_{\max}} \approx 23$, and this will be large enough for the relatively low Reynolds number solutions we seek here.

Problems for Lecture 9

1. An even function of x satisfies $f(-x) = f(x)$. An odd function of x satisfies $f(-x) = -f(x)$. Prove the following symmetry theorems:

a) If $f(x)$ is an even function of x, then $f'(x)$ is an odd function of x.

b) If $f(x)$ is an odd function of x, then $f'(x)$ is an even function of x.

Solutions to the Problems

Lecture 10 | Iteration equations

View this lecture on YouTube

The governing equations for a steady flow field may be written as

$$-\left(\frac{\partial^2\psi}{\partial\xi^2}+\frac{\partial^2\psi}{\partial\theta^2}\right)=e^{2\xi}\omega, \qquad -\left(\frac{\partial^2\omega}{\partial\xi^2}+\frac{\partial^2\omega}{\partial\theta^2}\right)=\frac{\mathrm{Re}}{2}\left(\frac{\partial\psi}{\partial\xi}\frac{\partial\omega}{\partial\theta}-\frac{\partial\psi}{\partial\theta}\frac{\partial\omega}{\partial\xi}\right).$$

We define $\psi_{i,j}=\psi(\xi_i,\theta_j)$ and $\omega_{i,j}=\omega(\xi_i,\theta_j)$ and use second-order centered difference approximations for all the derivatives. Application of the SOR method (*Numerical Methods for Engineers*, Lecture 66) results in the equations

$$\psi_{i,j}^{k+1}=(1-r_\psi)\psi_{i,j}^k+\frac{r_\psi}{4}\left(\psi_{i+1,j}^k+\psi_{i-1,j}^k+\psi_{i,j+1}^k+\psi_{i,j-1}^k+h^2e^{2\xi_i}\omega_{i,j}^k\right),$$
$$\omega_{i,j}^{k+1}=(1-r_\omega)\omega_{i,j}^k+\frac{r_\omega}{4}\left(\omega_{i+1,j}^k+\omega_{i-1,j}^k+\omega_{i,j+1}^k+\omega_{i,j-1}^k+\frac{\mathrm{Re}}{8}f_{i,j}^k\right),$$

where

$$f_{i,j}^k=\left(\psi_{i+1,j}^k-\psi_{i-1,j}^k\right)\left(\omega_{i,j+1}^k-\omega_{i,j-1}^k\right)-\left(\psi_{i,j+1}^k-\psi_{i,j-1}^k\right)\left(\omega_{i+1,j}^k-\omega_{i-1,j}^k\right).$$

Because these equations are nonlinear, iterations can easily become unstable. To maintain stability, the relaxation parameters, r_ψ and r_ω, may need to be less than one. Numerical experimentation may be necessary to obtain the best trade-off between computationally stability and computational speed. In addition, solutions at higher Reynolds numbers are best initialized by solutions at slightly lower Reynolds numbers.

The convergence of the iterations need to be monitored. We define

$$\varepsilon_\psi^{k+1}=\max_{i,j}\left|\psi_{i,j}^{k+1}-\psi_{i,j}^k\right|, \qquad \varepsilon_\omega^{k+1}=\max_{i,j}\left|\omega_{i,j}^{k+1}-\omega_{i,j}^k\right|.$$

Iterations can be stopped when the values of ε_ψ^{k+1} and ε_ω^{k+1} are less than some pre-defined error tolerance, say 10^{-8}.

Problems for Lecture 10

1. Derive the SOR finite difference equations for

a) the stream function equation,

$$-\left(\frac{\partial^2\psi}{\partial\xi^2}+\frac{\partial^2\psi}{\partial\theta^2}\right)=e^{2\xi}\omega;$$

b) the scalar vorticity equation,

$$-\left(\frac{\partial^2\omega}{\partial\xi^2}+\frac{\partial^2\omega}{\partial\theta^2}\right)=\frac{\mathrm{Re}}{2}\left(\frac{\partial\psi}{\partial\xi}\frac{\partial\omega}{\partial\theta}-\frac{\partial\psi}{\partial\theta}\frac{\partial\omega}{\partial\xi}\right).$$

Solutions to the Problems

Lecture 11 | Free-stream boundary conditions

View this lecture on YouTube

The dimensionless free-stream velocity field is given by $\boldsymbol{u} = \boldsymbol{i}$, which in polar coordinates is

$$\mathbf{u} = \cos\theta\hat{\mathbf{r}} - \sin\theta\hat{\boldsymbol{\theta}}.$$

Since $ru_r = \partial\psi/\partial\theta$ and $u_\theta = -\partial\psi/\partial r$, the stream function satisfies the free-stream conditions

$$\frac{\partial\psi}{\partial\theta} = r\cos\theta, \quad \frac{\partial\psi}{\partial r} = \sin\theta.$$

The solution of these equations can be seen to be

$$\psi(r,\theta) = r\sin\theta,$$

which also satisfies our boundary conditions $\psi = 0$ when $\theta = 0$ and π. On our log-polar coordinate grid, then, the free-stream boundary condition at $\xi = \xi_n$ is given by

$$\psi_{n,j} = e^{\xi_n}\sin\theta_j.$$

The free-stream flow field has zero vorticity, and the simplest, effective boundary condition on the scalar vorticity is

$$\omega_{n,j} = 0.$$

Problems for Lecture 11

1. Instead of assuming the scalar vorticity to be zero in the free stream, that is,

$$\omega_{n,j} = 0,$$

another option is to take the normal derivative to be zero, that is,

$$\left.\frac{\partial \omega}{\partial \xi}\right|_{(n,j)} = 0.$$

From second-order Taylor series expansions of $\omega_{n-1,j}$ and $\omega_{n-2,j}$, derive the corresponding boundary condition on $\omega_{n,j}$.

Solutions to the Problems

Lecture 12 | Cylinder boundary conditions

View this lecture on YouTube

Boundary conditions on the cylinder are determined from the no-penetration condition, $u_r = 0$, and no-slip condition, $u_\theta = 0$. And since $ru_r = \partial\psi/\partial\theta$ and $u_\theta = -\partial\psi/\partial r$, the stream function is constant and its normal derivative is zero. Since we know $\psi = 0$ at $y = 0$, the cylinder boundary conditions are

$$\psi_{1,j} = 0, \qquad \left.\frac{\partial\psi}{\partial\xi}\right|_{(1,j)} = 0.$$

The normal derivative boundary condition on the stream function will become a boundary condition on the vorticity. The scalar vorticity everywhere satisfies

$$\omega = -e^{-2\xi}\left(\frac{\partial^2\psi}{\partial\xi^2} + \frac{\partial^2\psi}{\partial\theta^2}\right),$$

and on the cylinder $\xi = 0$ and ψ is independent of θ so that

$$\omega_{1,j} = -\left.\frac{\partial^2\psi}{\partial\xi^2}\right|_{(1,j)}.$$

To obtain a boundary condition on the vorticity, we Taylor series expand the stream function one and two grid points away from the cylinder surface:

$$\psi_{2,j} = \psi_{1,j} + h\left.\frac{\partial\psi}{\partial\xi}\right|_{(1,j)} + \frac{1}{2}h^2\left.\frac{\partial^2\psi}{\partial\xi^2}\right|_{(1,j)} + \frac{1}{6}h^3\left.\frac{\partial^3\psi}{\partial\xi^3}\right|_{(1,j)} + \mathrm{O}(h^4),$$
$$\psi_{3,j} = \psi_{1,j} + 2h\left.\frac{\partial\psi}{\partial\xi}\right|_{(1,j)} + 2h^2\left.\frac{\partial^2\psi}{\partial\xi^2}\right|_{(1,j)} + \frac{4}{3}h^3\left.\frac{\partial^3\psi}{\partial\xi^3}\right|_{(1,j)} + \mathrm{O}(h^4).$$

Since the stream function and its normal derivative are zero on the cylinder surface, these Taylor series expansions reduce to

$$\psi_{2,j} = -\frac{1}{2}h^2\omega_{1,j} + \frac{1}{6}h^3\left.\frac{\partial^3\psi}{\partial\xi^3}\right|_{(1,j)} + \mathrm{O}(h^4), \qquad \psi_{3,j} = -2h^2\omega_{1,j} + \frac{4}{3}h^3\left.\frac{\partial^3\psi}{\partial\xi^3}\right|_{(1,j)} + \mathrm{O}(h^4).$$

After multiplying the first equation by eight and subtracting the second equation, we obtain

$$8\psi_{2,j} - \psi_{3,j} = -2h^2\omega_{1,j} + \mathrm{O}(h^4);$$

and the boundary condition on the vorticity, accurate to second-order, is given by

$$\omega_{1,j} = \frac{1}{2h^2}(\psi_{3,j} - 8\psi_{2,j}).$$

Problems for Lecture 12

1. The cylinder boundary condition on the scalar vorticity may be rewritten in terms of the stream function as

$$\left.\frac{\partial^2 \psi}{\partial \xi^2}\right|_{(1,j)} = \frac{1}{2h^2}(8\psi_{2,j} - \psi_{3,j}).$$

This boundary condition was derived using

$$\psi_{1,j} = 0, \qquad \left.\frac{\partial \psi}{\partial \xi}\right|_{(1,j)} = 0.$$

A stream function that is quadratic in ξ near the cylinder surface satisfies the above two conditions. Use it to test the validity of the derived boundary condition.

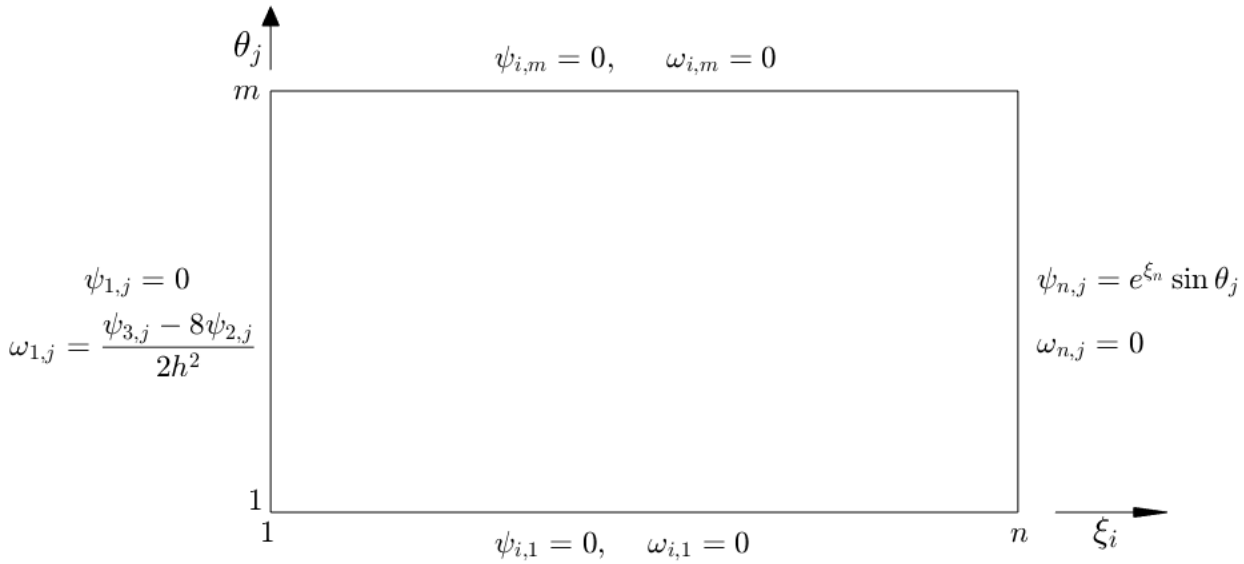
Solutions to the Problems

Lecture 13 | Summary of the boundary conditions

View this lecture on YouTube

We collect all the boundary conditions and summarize them here:

$$
\begin{aligned}
&\xi = 0, \quad 0 \le \theta \le \pi: && \psi_{1,j} = 0, \quad \omega_{1,j} = (\psi_{3,j} - 8\psi_{2,j})/2h^2;\\
&\xi = \xi_{\max}, \quad 0 \le \theta \le \pi: && \psi_{n,j} = e^{\xi_n} \sin\theta_j, \quad \omega_{n,j} = 0;\\
&\theta = 0, \quad 0 \le \xi \le \xi_{\max}: && \psi_{i,1} = 0, \quad \omega_{i,1} = 0;\\
&\theta = \pi, \quad 0 \le \xi \le \xi_{\max}: && \psi_{i,m} = 0, \quad \omega_{i,m} = 0.
\end{aligned}
$$



Problems for Lecture 13

1. Consider a grid with n and m gridpoints in the ξ and θ directions, respectively. Suppose the SOR iteration begins with the flow fields $\psi = \omega = 0$ everywhere except on the free-stream boundary, where

$$\psi_{n,j} = e^{\xi_n} \sin\theta_j.$$

How many SOR iterations does it take for ω to obtain its first nonzero value?

Solutions to the Problems

Lecture 14 | MATLAB program (steady) (Part A)

View this lecture on YouTube

Successful MATLAB programs will (1) calculate the stream function and scalar vorticity for the two-dimensional steady flow around a cylinder (simulation code), and; (2) visualize the solution (graphics code). We will provide students with a skeleton of the simulation code, and a finished graphics code. The solution for Re $= 10$ will be graded and students can perform additional calculations at other Reynolds numbers, if they so desire.

We have written the simulation code as a function, with the stream function `psi` and scalar vorticity `omega` as outputs, that is,

```
function [psi, omega] = flow_around_cylinder_steady
```

To make the calculation fast, we use only 101 grid points in both the ξ and θ directions ($101^2 = 10{,}201$ total grid points). The iteration is to be stopped when the absolute value of the change in `omega` or `psi` after one iteration is less than the error tolerance `delta=1.e-08` at every grid point.

The structure of the simulation code is as follows:

1. Define the grid.
2. Initialize the flow fields, including the free-stream boundary condition.
3. Set the relaxation parameters, error tolerance and any extra variables.
4. Perform the main SOR iteration loop:
   (a) Loop over all the grid points of the `psi` equation;
   (b) Set the `omega` boundary condition on the cylinder;
   (c) Loop over all the grid points of the `omega` equation;
   (d) Test for convergence of the fields, and continue the iteration or end the loop.
5. Plot the contours of the stream function.

Problems for Lecture 14

1. Using the SOR method, compute the steady solution for the stream function and scalar vorticity when $\mathrm{Re} = 10$.

Solutions to the Problems

Lecture 15 | MATLAB program (steady) (Part B)

View this lecture on YouTube

The graphics code that we provide plots the stream function for Re $= 10$. The stream function computation is on a grid in log-polar coordinates and the stream function needs to be plotted in Cartesian coordinates. Apart from the graphics itself, we will need to interpolate the `psi` values on the ξ-θ grid onto an x-y grid.

We make use of the MATLAB functions `meshgrid.m` and `interp2.m` to interpolate the stream function onto the Cartesian grid points `xi_i` and `theta_i`. The Cartesian grid is created using `meshgrid.m` and is given by the two arrays `X` and `Y`. The interpolation points in the ξ-θ space are given by

```
xi_i=0.5*log(X.^2+Y.^2);
theta_i=wrapTo2Pi(atan2(Y,X));
```

The graphics code is written as a function with stream function `psi` as input, that is,

```
function plot_Re10(psi)
```

The structure of the graphics code is as follows:

1. Define the ξ-θ grid.
2. Define the x-y grid.
3. Construct interpolation points.
4. Interpolate the values of `psi` onto the Cartesian grid.
5. Scale the interpolated values of `psi` for a better plot.
6. Set the color map.
7. Plot the color contours.
8. Set the contour line values.
9. Plot the contour lines.
10. Draw a black circle to represent the cylinder.
11. Beautify the plot.

Problems for Lecture 15

1. Study the graphics code and try to understand it. You may need to Google some unfamiliar MATLAB functions.

Solutions to the Problems

Week III

Unsteady Flows

In this week's lectures, we will solve for a time-dependent stream function and scalar vorticity. Our goal is to simulate the famous Kármán street, where periodic vortices are shed from the top and bottom of the cylinder. Applying periodic boundary conditions in the polar angle θ, we use a second-order finite difference method to discretize the equations in space. The stream-function equation is a system of linear equations and can be solved by a direct method using the Matlab *backslash operator. To increase the speed of the computation, we make use of the LU decomposition. The time integration of the vorticity equation is performed using the* Matlab *function* `ode23.m`.

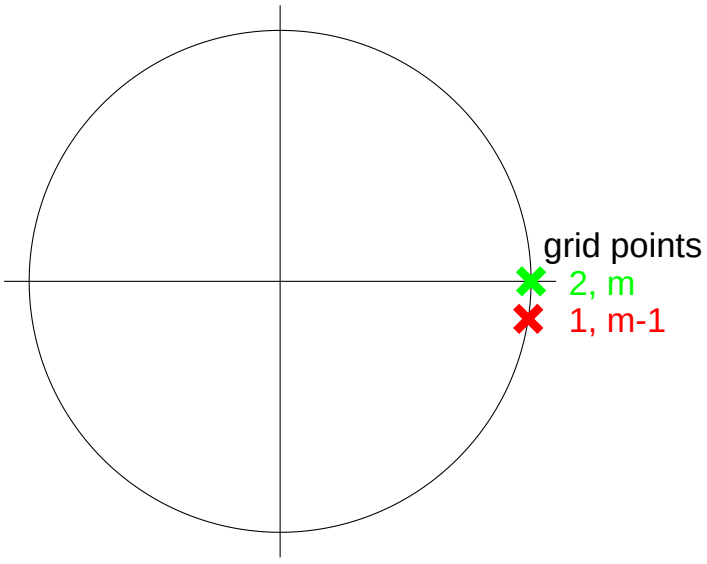
For the computational project, students will flesh out the skeleton of a code that time-advances the vorticity field. Experiments have shown that the steady solution becomes unstable for $Re < 46$ and we ask students to perform a calculation at $Re = 60$. After building a program that passes the Matlab *Grader, students can then write code to create a movie file of the Kármán street.*

Lecture 16 | Periodic boundary conditions

View this lecture on YouTube

In the unsteady flow, vortices are shed above and below the cylinder. We will need to solve for the flow over the entire range of the polar angle, $0 \leq \theta < 2\pi$.

We implement periodic boundary conditions. Let m be the total number of grid points in the θ angle. We define the first grid point to be one grid cell below zero, and the last grid point to be 2π (see figure).



The polar angle is now discretized as

$$\theta_j = (j-2)h, \qquad j = 1, 2, \ldots, m,$$

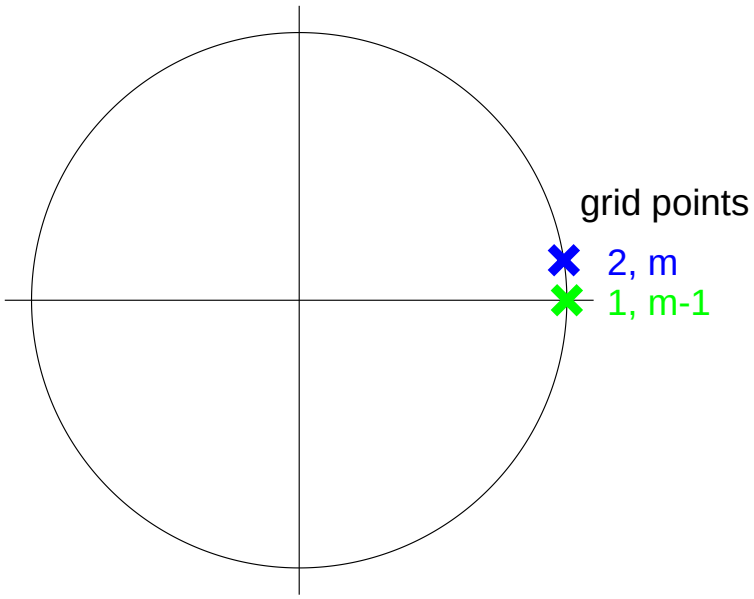
and since $\theta_m = 2\pi$, we have $h = 2\pi/(m-2)$.

The first and last grid points are duplicates of the $(m-1)$ and second internal grid points and are called ghost points. Periodic boundary conditions then requires for these ghost points

$$\psi_{i,1} = \psi_{i,m-1}, \quad \omega_{i,1} = \omega_{i,m-1}; \qquad \psi_{i,m} = \psi_{i,2}, \quad \omega_{i,m} = \omega_{i,2}.$$

Problems for Lecture 16

1. Our implementation of periodic boundary conditions in the polar angle uses two extra ghost points: one point below the x-axis and one point on the x-axis. It is equally fine to implement periodic boundary conditions with one point above the x-axis and one point on the x-axis, as shown in the figure below.



Write down the two appropriate periodic boundary conditions for the first and last grid points. What is the difference between this implementation and the one in lecture?

Solutions to the Problems

Lecture 17 | Finite difference equations

View this lecture on YouTube

If we know the vorticity ω at time t, we can solve a Poisson equation for the stream function,

$$-\left(\frac{\partial^2\psi}{\partial\xi^2}+\frac{\partial^2\psi}{\partial\theta^2}\right)=e^{2\xi}\omega.$$

Then with both the stream function and vorticity known at time t, we can time integrate the vorticity equation,

$$\frac{\partial\omega}{\partial t}=\frac{2e^{-2\xi}}{\mathrm{Re}}\left(\frac{\partial^2\omega}{\partial\xi^2}+\frac{\partial^2\omega}{\partial\theta^2}\right)+e^{-2\xi}\left(\frac{\partial\psi}{\partial\xi}\frac{\partial\omega}{\partial\theta}-\frac{\partial\psi}{\partial\theta}\frac{\partial\omega}{\partial\xi}\right).$$

We discretize the spatial derivatives as before, using a second-order finite difference scheme. The stream-function equation is

$$4\psi_{i,j}-\psi_{i+1,j}-\psi_{i-1,j}-\psi_{i,j+1}-\psi_{i,j-1}=h^2e^{2\xi_i}\omega_{i,j},$$

and the vorticity equation becomes

$$\begin{aligned}\frac{d\omega_{i,j}}{dt}=&\frac{2e^{-2\xi_i}}{h^2\mathrm{Re}}\left(\omega_{i+1,j}+\omega_{i-1,j}+\omega_{i,j+1}+\omega_{i,j-1}-4\omega_{i,j}\right)\\&+\frac{e^{-2\xi_i}}{4h^2}\left[\left(\psi_{i+1,j}-\psi_{i-1,j}\right)\left(\omega_{i,j+1}-\omega_{i,j-1}\right)-\left(\psi_{i,j+1}-\psi_{i,j-1}\right)\left(\omega_{i+1,j}-\omega_{i-1,j}\right)\right].\end{aligned}$$

In the next few lectures, we discuss how to solve these two coupled equations.

Problems for Lecture 17

1. Derive the finite difference equation for the scalar vorticity from the time-dependent vorticity equation,

$$\frac{\partial \omega}{\partial t} = \frac{2e^{-2\xi}}{\mathrm{Re}} \left(\frac{\partial^2 \omega}{\partial \xi^2} + \frac{\partial^2 \omega}{\partial \theta^2} \right) + e^{-2\xi} \left(\frac{\partial \psi}{\partial \xi} \frac{\partial \omega}{\partial \theta} - \frac{\partial \psi}{\partial \theta} \frac{\partial \omega}{\partial \xi} \right).$$

Solutions to the Problems

Lecture 18 | Stream-function computation

View this lecture on YouTube

The discrete stream-function equation is given by

$$4\psi_{i,j} - \psi_{i+1,j} - \psi_{i-1,j} - \psi_{i,j+1} - \psi_{i,j-1} = h^2 e^{2\xi_i} \omega_{i,j}.$$

We define the right-hand side of this equation as

$$\tilde{\omega}_{i,j} = h^2 e^{2\xi_i} \omega_{i,j},$$

and apply natural ordering [Numerical Methods for Engineers, Lecture 63] to map $(i, j) \to k = i + (j-1)n$. The stream-function equation transforms to

$$4\psi_k - \psi_{k+1} - \psi_{k-1} - \psi_{k+n} - \psi_{k-n} = \tilde{\omega}_k.$$

This is a matrix equation $\mathrm{A}\psi = \mathrm{b}$, where each value of k corresponds to a row of the matrix A. The matrix A has dimension mn-by-mn and has five diagonal bands: a four on the main diagonal, and a negative one on one and n above and below the main diagonal. These diagonal bands extend only over the rows in which k indexes an interior point of the domain.

When k indexes a boundary point, however, the discrete stream function equation no longer applies and the corresponding rows of the matrix A and right-hand side b are modified to enforce the boundary conditions. We discuss the boundary conditions in the next lecture.

Problems for Lecture 18

1. Consider the calculation of $\tilde{\omega}_k$ in MATLAB, where

$$\tilde{\omega}_{i,j} = h^2 e^{2\xi_i} \omega_{i,j},$$

the dimension of ω is n-by-m, and we have applied natural ordering to map $(i, j) \to k = i + (j-1)n$.

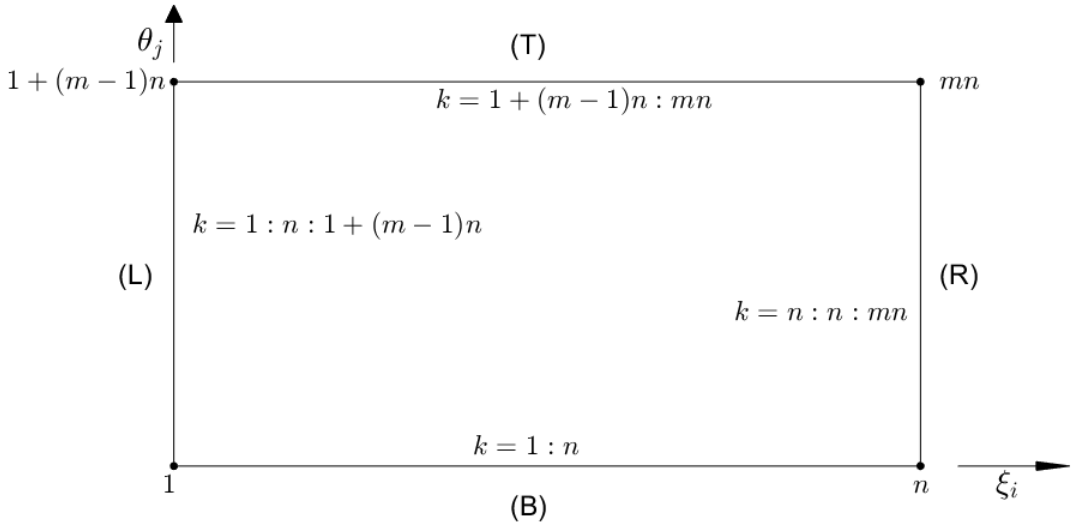
Map the mathematical variables above into the following MATLAB variables: `omega_tilde`, `h`, `xi`, `omega`, `n`, `m`. Write the MATLAB code that computes the `mn` sized vector `omega_tilde` given that all the other variables have previously been defined.

Solutions to the Problems

Lecture 19 | Stream-function boundary conditions

View this lecture on YouTube

We want to solve the matrix equation $\mathrm{A}\psi = \mathrm{b}$ and consider now the rows of A corresponding to points on the boundaries of the computational domain. The values of k on the boundaries are shown in the figure below, where we borrow the colon notation from MATLAB. We label the four sides: bottom (B), top (T), left (L) and right (R). On (L) where $\xi = 0$, we have $\psi_{1,j} = 0$; and on (R) where $\xi = \xi_n$,

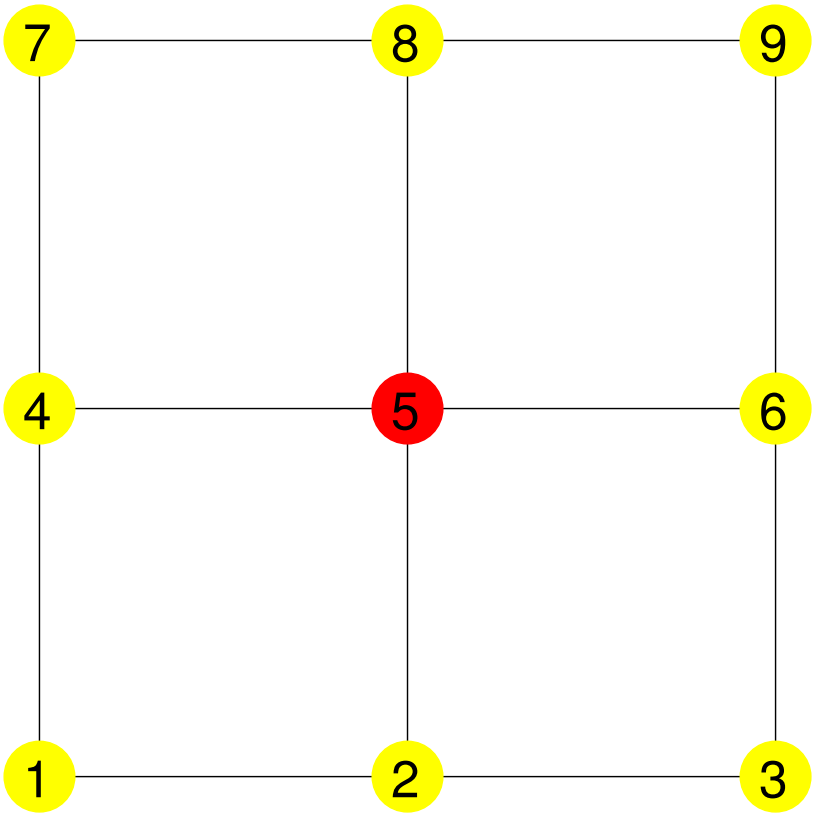


we have the free-stream condition $\psi_{n,j} = e^{\xi_n} \sin\theta_j$. These boundary conditions are implemented by replacing the kth row of A by the corresponding kth row of the $mn \times mn$ identity matrix, and setting the right-hand side row element either to zero or to the free-stream condition.

Periodic boundary conditions are implemented on (B) and (T). On (B), we implement $\psi_{i,1} - \psi_{i,m-1} = 0$. That is, in rows $k = 1{:}n$ of the matrix A, we place the difference of two rows of the identity matrix: rows $1{:}n$ minus rows $1 + (m-2)n{:}(m-1)n$. The corresponding row elements in b are set to zero. On (T), we implement $\psi_{i,m} - \psi_{i,2} = 0$. That is, in rows $k = 1 + (m-1)n{:}mn$ of the matrix A, we place the difference of two rows of the identity matrix: rows $1 + (m-1)n{:}mn$ minus rows $(1+n){:}2n$. And again the corresponding row elements in b are set to zero.

Problems for Lecture 19

1. Construct the nine-by-nine matrix equation for the stream function on the three-by-three grid shown below, where the k indexing is labeled. Note that only grid point five is an internal point.



Solutions to the Problems

Lecture 20 | Vorticity computation

View this lecture on YouTube

The discrete scalar vorticity equation at interior grid points is given by

$$\frac{d\omega_{i,j}}{dt} = \frac{2e^{-2\xi_i}}{h^2\mathrm{Re}}\left(\omega_{i+1,j} + \omega_{i-1,j} + \omega_{i,j+1} + \omega_{i,j-1} - 4\omega_{i,j}\right) + \frac{e^{-2\xi_i}}{4h^2}\left[\left(\psi_{i+1,j} - \psi_{i-1,j}\right)\left(\omega_{i,j+1} - \omega_{i,j-1}\right) - \left(\psi_{i,j+1} - \psi_{i,j-1}\right)\left(\omega_{i+1,j} - \omega_{i-1,j}\right)\right]$$

This equation is a system of $(n-2)(m-2)$ first-order differential equations. The stream function and scalar vorticity are assumed to be known at time t, and this system of equations is integrated to solve for the scalar vorticity at time $t + \Delta t$.

Although we are solving only for the scalar vorticity at the interior points, the right-hand side requires the values of the stream function and scalar vorticity at the boundaries. Our matrix equation for the stream function already includes boundary conditions, and the boundary conditions on the scalar vorticity are

$$\omega_{1,j} = \frac{\psi_{3,j} - 8\psi_{2,j}}{2h^2}, \quad \omega_{n,j} = 0, \quad \omega_{i,1} = \omega_{i,m-1} \quad \omega_{i,m} = \omega_{i,2}.$$

To integrate the system of first-order differential equations, a MATLAB ode solver can be used. We have timed the solvers `ode45.m`, `ode23.m` and `ode113.m` using the default error tolerance `RelTol = 1.e-3`, and the most commonly used solver, `ode45.m`, appears to require about 50% more derivative evaluations than the other two solvers. We have therefore chosen to use `ode23.m` in our program.

Problems for Lecture 20

1. In an efficiently written MATLAB program, often one line of code takes up most of the computational time. In the unsteady flow problem, it will certainly be the call to `ode23.m`. But within the function that defines the time derivative of the vorticity field, which line of code do you think will require most of the computational time?

Solutions to the Problems

Lecture 21 | MATLAB program (unsteady) (Part A)

View this lecture on YouTube

To compute the Kármán vortex street, one can first compute a time series of the scalar vorticity in the unsteady flow regime, and second, create images of the vorticity field and add them to a movie file. For the assessment, we will only ask students to fill out the skeleton of a code to time-advance the vorticity field. Students, on their own, can create a time series of fields and produce a movie of the Kármán street, as explained in the next Lecture.

The code template we give students uses 101 grid points in the ξ direction and 202 grid points in the θ direction. Dimensionless time is to be advanced to only $t = 0.5$, and the resulting vorticity field will be graded. We found that a near steady flow field develops by $t = 50$, after which a growing instability can be seen visually near $t = 900$, with a fully formed Kármán street at $t = 1100$.

The structure of the simulation code for the assessment is as follows:

1. Define the grid and time-stepping parameters.
2. Initialize the flow fields, including the free-stream boundary condition.
3. Construct the matrix A for the ψ equation and compute A = LU.
4. Compute and save time-independent factors.
5. Advance the vorticity field using `ode23.m`.
6. Compute the stream function and obtain the vorticity boundary values.
9. Plot the vorticity field.

The function called by `ode23.m` has the following structure:

1. Compute the stream function.
2. Set the boundary conditions on the vorticity field.
3. Compute the time-derivative of the vorticity.

Problems for Lecture 21

1. Using `ode23.m`, compute the unsteady solution for the scalar vorticity when $\mathrm{Re} = 60$.

Solutions to the Problems

Lecture 22 | MATLAB program (unsteady) (Part B)

View this lecture on YouTube

After you have a code that computes a time-dependent vorticity field, you might want to create a movie file of the Kármán vortex street. One could add movie frames on the fly as you compute the fields. But since most of the computational time is spent computing the fields, and often we need to tweak the graphics, a better solution is to write a time series of fields into `.mat` files. These fields can then be loaded into another program that creates the movie file.

To write the `.mat` files, we define a file name such as

```
output_file='.\fields\Re_60_';
```

where we choose to write the fields into a subdirectory of our current working directory. We then loop over the call to `ode23.m`, advance the vorticity field over a fixed time interval, and save each computed vorticity field to disk using the MATLAB function call

```
save([output_file,num2str(k),'.mat'],'omega');
```

where `k` is the loop variable, for instance.

The separate graphics program then reads in the fields one-by-one, and plots the vorticity using the MATLAB function `imagesc`. To initialize the movie file, we use

```
writerObj = VideoWriter('movie.mp4','MPEG-4');
writerObj.Quality = 100;
writerObj.FrameRate=24;
open(writerObj);
```

To capture movie frames and save them into a movie file, we use the commands

```
frame=getframe(gcf);
writeVideo(writerObj,frame);
```

Finally, we conclude the program with the command

```
close(writerObj);
```

I will provide you a copy of my graphics code, but feel free to modify it to suit your desired outcome and taste.

Problems for Lecture 22

1. Study the graphics code that creates the movie file and try to understand what it does.

Solutions to the Problems

Problem solutions and MATLAB learner templates

Solutions to the Problems for Lecture 1

1. The first component of the Navier-Stokes equation reduces to

$$\nu \frac{d^2u}{dy^2} = 0.$$

The boundary conditions on the plates are $u(0) = 0$ and $u(d) = U$. We find $u(y) = Uy/d$, so that

$$\boldsymbol{u} = \frac{Uy}{d}\boldsymbol{i}.$$

2. The first component of the Navier-Stokes equation reduces to

$$-\frac{1}{\rho}\frac{dp}{dx} + \nu \frac{d^2u}{dy^2} = 0.$$

Using $dp/dx = -G$ leads to

$$\frac{d^2u}{dy^2} = -\frac{G}{\nu\rho},$$

which can be solved using the no-slip boundary conditions $u(0) = u(d) = 0$. We find

$$\boldsymbol{u} = \frac{Gd^2}{2\nu\rho}\left(\frac{y}{d}\right)\left(1 - \frac{y}{d}\right)\boldsymbol{i}.$$

3. The first component of the Navier-Stokes equation becomes

$$-\frac{1}{\rho}\frac{dp}{dx} + \nu\left(\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) = 0.$$

Using $dp/dx = -G$ leads to

$$\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = -\frac{G}{\nu\rho}.$$

We use polar coordinates in the y-z plane. With $r = \sqrt{y^2 + z^2}$, and the Laplacian written in polar coordinates, the differential equation for $u = u(r)$ then becomes

$$\frac{d}{dr}\left(r\frac{du}{dr}\right) = -\frac{Gr}{\nu\rho},$$

with no-slip boundary condition $u(R) = 0$. The first integration from 0 to r yields

$$r\frac{du}{dr} = -\frac{Gr^2}{2\nu\rho};$$

and after division by r, the second integration from r to R yields

$$u(r) = \frac{GR^2}{4\nu\rho}\left(1 - \left(\frac{r}{R}\right)^2\right),$$

so that

$$\boldsymbol{u} = \frac{GR^2}{4\nu\rho}\left(1 - \frac{y^2 + z^2}{R^2}\right)\boldsymbol{i}.$$

Solutions to the Problems for Lecture 2

1.

a) We prove that $\boldsymbol{\nabla} \times \{(\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u}\} = \boldsymbol{\nabla} \times (\boldsymbol{\omega} \times \boldsymbol{u})$. To begin, we first prove the identity

$$\boldsymbol{u} \times (\boldsymbol{\nabla} \times \boldsymbol{u}) = \frac{1}{2}\boldsymbol{\nabla}(\boldsymbol{u} \cdot \boldsymbol{u}) - (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u}.$$

Proof proceeds by considering the ith component of the left-hand side:

$$\begin{aligned}
[\boldsymbol{u} \times (\boldsymbol{\nabla} \times \boldsymbol{u})]_i &= \epsilon_{ijk} u_j \epsilon_{klm} \frac{\partial u_m}{\partial x_l} && \text{(curl in component notation)} \\
&= \epsilon_{kij}\epsilon_{klm} u_j \frac{\partial u_m}{\partial x_l} && (\epsilon_{ijk} = \epsilon_{kij}) \\
&= (\delta_{il}\delta_{jm} - \delta_{im}\delta_{jl}) u_j \frac{\partial u_m}{\partial x_l} && (\epsilon_{kij}\epsilon_{klm} = \delta_{il}\delta_{jm} - \delta_{im}\delta_{jl}) \\
&= u_j \frac{\partial u_j}{\partial x_i} - u_j \frac{\partial u_i}{\partial x_j} && (\delta_{ij} A_j = A_i) \\
&= \frac{1}{2}\frac{\partial}{\partial x_i}(u_j u_j) - u_j \frac{\partial u_i}{\partial x_j} \\
&= \left[\frac{1}{2}\boldsymbol{\nabla}(\boldsymbol{u} \cdot \boldsymbol{u})\right]_i - [(\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u}]_i . && \text{(back to vector notation)}
\end{aligned}$$

Therefore,

$$(\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u} = \frac{1}{2}\boldsymbol{\nabla}(\boldsymbol{u} \cdot \boldsymbol{u}) - \boldsymbol{u} \times (\boldsymbol{\nabla} \times \boldsymbol{u}),$$

and

$$\begin{aligned}
\boldsymbol{\nabla} \times (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u} &= \boldsymbol{\nabla} \times \left(\frac{1}{2}\boldsymbol{\nabla}(\boldsymbol{u} \cdot \boldsymbol{u}) - \boldsymbol{u} \times (\boldsymbol{\nabla} \times \boldsymbol{u})\right) \\
&= \frac{1}{2}\boldsymbol{\nabla} \times (\boldsymbol{\nabla}(\boldsymbol{u} \cdot \boldsymbol{u})) - \boldsymbol{\nabla} \times (\boldsymbol{u} \times (\boldsymbol{\nabla} \times \boldsymbol{u})) && \text{(distribution law for curl)} \\
&= -\boldsymbol{\nabla} \times (\boldsymbol{u} \times (\boldsymbol{\nabla} \times \boldsymbol{u})) && \text{(curl of a gradient equals zero)} \\
&= -\boldsymbol{\nabla} \times (\boldsymbol{u} \times \boldsymbol{\omega}) && (\boldsymbol{\nabla} \times \boldsymbol{u} = \boldsymbol{\omega}) \\
&= \boldsymbol{\nabla} \times (\boldsymbol{\omega} \times \boldsymbol{u}). && (\boldsymbol{u} \times \boldsymbol{\omega} = -\boldsymbol{\omega} \times \boldsymbol{u})
\end{aligned}$$

b) We prove that

$$\boldsymbol{\nabla} \times (\boldsymbol{\omega} \times \boldsymbol{u}) = (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{\omega} - (\boldsymbol{\omega} \cdot \boldsymbol{\nabla})\boldsymbol{u},$$

using the vector identity

$$\boldsymbol{\nabla} \times (\boldsymbol{u} \times \boldsymbol{v}) = \boldsymbol{u}(\boldsymbol{\nabla} \cdot \boldsymbol{v}) - \boldsymbol{v}(\boldsymbol{\nabla} \cdot \boldsymbol{u}) + (\boldsymbol{v} \cdot \boldsymbol{\nabla})\boldsymbol{u} - (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{v}.$$

We have

$$\begin{aligned}
\boldsymbol{\nabla} \times (\boldsymbol{\omega} \times \boldsymbol{u}) &= \boldsymbol{\omega}(\boldsymbol{\nabla} \cdot \boldsymbol{u}) - \boldsymbol{u}(\boldsymbol{\nabla} \cdot \boldsymbol{\omega}) + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{\omega} - (\boldsymbol{\omega} \cdot \boldsymbol{\nabla})\boldsymbol{u} && \text{(vector identity)} \\
&= (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{\omega} - (\boldsymbol{\omega} \cdot \boldsymbol{\nabla}). && (\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0, \quad \boldsymbol{\nabla} \cdot \boldsymbol{\omega} = 0)
\end{aligned}$$

Solutions to the Problems for Lecture 3

1. The cylinder boundary in Cartesian coordinates is given by

$$x^2 + y^2 = R^2,$$

and in polar coordinates is given by

$$r = R.$$

The free stream velocity in Cartesian coordinates is given by

$$\boldsymbol{u}_f = U\boldsymbol{i},$$

and in polar coordinates is given by

$$\boldsymbol{u}_f = U(\cos\theta\,\hat{r} - \sin\theta\,\hat{\theta}).$$

Solutions to the Problems for Lecture 4

1. We have

$$\boldsymbol{\omega} \times \boldsymbol{u} = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ 0 & 0 & \omega \\ u & v & 0 \end{vmatrix} = -v\omega\boldsymbol{i} + u\omega\boldsymbol{j},$$

and

$$\begin{aligned} \boldsymbol{\nabla} \times (\boldsymbol{\omega} \times \boldsymbol{u}) &= \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ \partial/\partial x & \partial/\partial y & 0 \\ -v\omega & u\omega & 0 \end{vmatrix} = \left(\frac{\partial}{\partial x}(u\omega) + \frac{\partial}{\partial y}(v\omega)\right)\boldsymbol{k} \\ &= \left[\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)\omega + \left(u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y}\right)\right]\boldsymbol{k} \\ &= \left(u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y}\right)\boldsymbol{k}. \end{aligned}$$

2.

$$\boldsymbol{\omega} = \boldsymbol{\nabla} \times \boldsymbol{u} = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ \partial/\partial x & \partial/\partial y & 0 \\ u & v & 0 \end{vmatrix} = \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right)\boldsymbol{k}.$$

The scalar vorticity is then defined as

$$\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}.$$

3. In Cartesian coordinates, the two-dimensional scalar vorticity equation is written as

$$\frac{\partial \omega}{\partial t} + u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y} = \nu\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right).$$

Solutions to the Problems for Lecture 5

1. The two-dimensional incompressibility condition in Cartesian coordinates is given by

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0.$$

The appropriate definition of the stream function $\psi = \psi(x, y)$ that satisfies this condition is

$$u = \frac{\partial \psi}{\partial y}, \qquad v = -\frac{\partial \psi}{\partial x}.$$

2. The scalar vorticity in Cartesian coordinates is given by

$$\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}.$$

Using the definition of the stream function, we have

$$\omega = -\left(\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2}\right)$$

3. Since $\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0$ we can write $\boldsymbol{u} = \boldsymbol{\nabla} \times \boldsymbol{A}$, where $\boldsymbol{A}$ is called a vector potential. In two dimensions, we have

$$u\boldsymbol{i} + v\boldsymbol{j} = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ \partial/\partial x & \partial/\partial y & 0 \\ A_1 & A_2 & A_3 \end{vmatrix} = \frac{\partial A_3}{\partial y}\boldsymbol{i} - \frac{\partial A_3}{\partial x}\boldsymbol{j} + \left(\frac{\partial A_2}{\partial x} - \frac{\partial A_1}{\partial y}\right)\boldsymbol{k}.$$

Therefore,

$$u = \frac{\partial A_3}{\partial y}, \qquad v = -\frac{\partial A_3}{\partial x},$$

and the stream function can be identified as the third component of the vector potential $\boldsymbol{A}$.

Solutions to the Problems for Lecture 6

1. In Cartesian coordinates, $d\boldsymbol{r} = dx\boldsymbol{i} + dy\boldsymbol{j}$, and with the velocity field parallel to $d\boldsymbol{r}$, we have

$$0 = \boldsymbol{u} \times d\boldsymbol{r} = (u\boldsymbol{i} + v\boldsymbol{j}) \times (dx\boldsymbol{i} + dy\boldsymbol{j}) = (-vdx + udy)\boldsymbol{k}.$$

But since the stream function ψ is defined by

$$u = \frac{\partial \psi}{\partial y}, \qquad v = -\frac{\partial \psi}{\partial x},$$

we have determined that

$$0 = \left(\frac{\partial \psi}{\partial x}dx + \frac{\partial \psi}{\partial y}dy\right)\boldsymbol{k} = d\psi\boldsymbol{k},$$

or that the stream function is constant along streamlines.

Solutions to the Problems for Lecture 7

1. The nondimensional vorticity equation can be written as

$$\nabla^2\omega = \frac{\mathrm{Re}}{2}\left(\frac{\partial\omega}{\partial t} + \frac{1}{r}\left(\frac{\partial\psi}{\partial\theta}\frac{\partial\omega}{\partial r} - \frac{\partial\psi}{\partial r}\frac{\partial\omega}{\partial\theta}\right)\right);$$

and if we set $\mathrm{Re} = 0$, we obtain the Laplace equation $\nabla^2\omega = 0$. Since $\nabla^2\psi = -\omega$, taking the Laplacian of both sides results in the biharmonic equation, usually written as

$$\nabla^4\psi = 0.$$

The attempt to solve this equation for the flow around an infinite circular cylinder leads to Stokes' paradox.

Solutions to the Problems for Lecture 8

1. The time-dependent governing equations are given by

$$\left(\frac{\partial^2\psi}{\partial\xi^2} + \frac{\partial^2\psi}{\partial\theta^2}\right) = -e^{2\xi}\omega,$$
$$\frac{\partial\omega}{\partial t} + e^{-2\xi}\left(\frac{\partial\psi}{\partial\theta}\frac{\partial\omega}{\partial\xi} - \frac{\partial\psi}{\partial\xi}\frac{\partial\omega}{\partial\theta}\right) = \frac{2}{\mathrm{Re}}e^{-2\xi}\left(\frac{\partial^2\omega}{\partial\xi^2} + \frac{\partial^2\omega}{\partial\theta^2}\right).$$

In a steady flow, we have $\partial\omega/\partial t = 0$. The governing equations can then be written neatly as

$$\left(\frac{\partial^2\psi}{\partial\xi^2} + \frac{\partial^2\psi}{\partial\theta^2}\right) = -e^{2\xi}\omega, \qquad \left(\frac{\partial^2\omega}{\partial\xi^2} + \frac{\partial^2\omega}{\partial\theta^2}\right) = \frac{\mathrm{Re}}{2}\left(\frac{\partial\psi}{\partial\theta}\frac{\partial\omega}{\partial\xi} - \frac{\partial\psi}{\partial\xi}\frac{\partial\omega}{\partial\theta}\right).$$

Solutions to the Problems for Lecture 9

1.

a) Suppose $f(x)$ is an even function of x so that $f(-x) = f(x)$. Then using the definition of the derivative,

$$\begin{aligned} f'(-x) &= \lim_{h\to 0}\frac{f(-x+h) - f(-x)}{h} \\ &= \lim_{h\to 0}\frac{f(x-h) - f(x)}{h} \\ &= -\lim_{h\to 0}\frac{f(x) - f(x-h)}{h} \\ &= -f'(x). \end{aligned}$$

b) Suppose $f(x)$ is an odd function of x so that $f(-x) = -f(x)$. Then using the definition of the

derivative

$$\begin{aligned} f'(-x) &= \lim_{h\to 0} \frac{f(-x+h)-f(-x)}{h} \\ &= \lim_{h\to 0} \frac{-f(x-h)+f(x)}{h} \\ &= \lim_{h\to 0} \frac{f(x)-f(x-h)}{h} \\ &= f'(x). \end{aligned}$$

Solutions to the Problems for Lecture 10

1.

a) The stream function equation is given by

$$-\left(\frac{\partial^2\psi}{\partial\xi^2}+\frac{\partial^2\psi}{\partial\theta^2}\right)=e^{2\xi}\omega.$$

The second-order finite difference approximation to the second partial derivatives yields

$$-\left(\frac{\psi_{i+1,j}-2\psi_{i,j}+\psi_{i-1,j}}{h^2}+\frac{\psi_{i,j+1}-2\psi_{i,j}+\psi_{i,j-1}}{h^2}\right)=e^{2\xi_i}\omega_{i,j}$$

We multiply by h^2 and collect terms to obtain

$$4\psi_{i,j}-\psi_{i+1,j}-\psi_{i-1,j}-\psi_{i,j+1}-\psi_{i,j-1}=h^2e^{2\xi_i}\omega_{i,j}$$

To construct the SOR method, we rewrite this equation as

$$\psi_{i,j}=\psi_{i,j}+\frac{1}{4}\left(\psi_{i+1,j}+\psi_{i-1,j}+\psi_{i,j+1}+\psi_{i,j-1}-4\psi_{i,j}+h^2e^{2\xi_i}\omega_{i,j}\right).$$

We make the left-hand side the $(k+1)$-th iteration and the right-hand side the k-th iteration, and we adjust the correction to $\psi_{i,j}$ by a relaxation factor r_ψ. The result becomes

$$\begin{aligned} \psi_{i,j}^{k+1} &= \psi_{i,j}^{k}+\frac{r_\psi}{4}\left(\psi_{i+1,j}^{k}+\psi_{i-1,j}^{k}+\psi_{i,j+1}^{k}+\psi_{i,j-1}^{k}-4\psi_{i,j}^{k}+h^2e^{2\xi_i}\omega_{i,j}^{k}\right) \\ &= (1-r_\psi)\psi_{i,j}^{k}+\frac{r_\psi}{4}\left(\psi_{i+1,j}^{k}+\psi_{i-1,j}^{k}+\psi_{i,j+1}^{k}+\psi_{i,j-1}^{k}+h^2e^{2\xi_i}\omega_{i,j}^{k}\right). \end{aligned}$$

b) The scalar vorticity equation is given by

$$-\left(\frac{\partial^2\omega}{\partial\xi^2}+\frac{\partial^2\omega}{\partial\theta^2}\right)=\frac{\mathrm{Re}}{2}\left(\frac{\partial\psi}{\partial\xi}\frac{\partial\omega}{\partial\theta}-\frac{\partial\psi}{\partial\theta}\frac{\partial\omega}{\partial\xi}\right).$$

Second-order finite difference approximations for the first- and second-derivatives yields

$$\begin{aligned} &-\left(\frac{\omega_{i+1,j}-2\omega_{i,j}+\omega_{i-1,j}}{h^2}+\frac{\omega_{i,j+1}-2\omega_{i,j}+\omega_{i,j-1}}{h^2}\right) \\ &=\frac{\mathrm{Re}}{2}\left[\left(\frac{\psi_{i+1,j}-\psi_{i-1,j}}{2h}\right)\left(\frac{\omega_{i,j+1}-\omega_{i,j-1}}{2h}\right)-\left(\frac{\psi_{i,j+1}-\psi_{i,j-1}}{2h}\right)\left(\frac{\omega_{i+1,j}-\omega_{i-1,j}}{2h}\right)\right]. \end{aligned}$$

Multiplying by h^2 and collecting terms yields

$$\begin{aligned}&4\omega_{i,j} - \omega_{i+1,j} - \omega_{i-1,j} - \omega_{i,j+1} - \omega_{i,j-1}\\ &= \frac{\mathrm{Re}}{8}\left[\left(\psi_{i+1,j} - \psi_{i-1,j}\right)\left(\omega_{i,j+1} - \omega_{i,j-1}\right) - \left(\psi_{i,j+1} - \psi_{i,j-1}\right)\left(\omega_{i+1,j} - \omega_{i-1,j}\right)\right]\end{aligned}$$

For notational clarity, we define

$$f_{i,j} = \left(\psi_{i+1,j} - \psi_{i-1,j}\right)\left(\omega_{i,j+1} - \omega_{i,j-1}\right) - \left(\psi_{i,j+1} - \psi_{i,j-1}\right)\left(\omega_{i+1,j} - \omega_{i-1,j}\right).$$

We then rewrite the equation as

$$\omega_{i,j} = \omega_{i,j} + \frac{1}{4}\left(\omega_{i+1,j} + \omega_{i-1,j} + \omega_{i,j+1} + \omega_{i,j-1} - 4\omega_{i,j} + \frac{\mathrm{Re}}{8} f_{i,j}\right).$$

Again, we make the left-hand side the $(k+1)$-th iteration and the right-hand side the k-th iteration, and we adjust the correction to $\omega_{i,j}$ by a relaxation factor r_ω. The result becomes

$$\begin{aligned}\omega_{i,j}^{k+1} &= \omega_{i,j}^{k} + \frac{r_\omega}{4}\left(\omega_{i+1,j}^{k} + \omega_{i-1,j}^{k} + \omega_{i,j+1}^{k} + \omega_{i,j-1}^{k} - 4\omega_{i,j}^{k} + \frac{\mathrm{Re}}{8} f_{i,j}^{k}\right)\\ &= (1 - r_\omega)\omega_{i,j}^{k} + \frac{r_\omega}{4}\left(\omega_{i+1,j}^{k} + \omega_{i-1,j}^{k} + \omega_{i,j+1}^{k} + \omega_{i,j-1}^{k} + \frac{\mathrm{Re}}{8} f_{i,j}^{k}\right),\end{aligned}$$

with

$$f_{i,j}^{k} = \left(\psi_{i+1,j}^{k} - \psi_{i-1,j}^{k}\right)\left(\omega_{i,j+1}^{k} - \omega_{i,j-1}^{k}\right) - \left(\psi_{i,j+1}^{k} - \psi_{i,j-1}^{k}\right)\left(\omega_{i+1,j}^{k} - \omega_{i-1,j}^{k}\right).$$

Solutions to the Problems for Lecture 11

1. We Taylor series expand the scalar vorticity one and two grid points inside the free-stream boundary. That is,

$$\begin{aligned}\omega_{n-1,j} &= \omega_{n,j} - h\left.\frac{\partial \omega}{\partial \xi}\right|_{(n,j)} + \frac{h^2}{2}\left.\frac{\partial^2 \omega}{\partial \xi^2}\right|_{(n,j)} + \mathrm{O}(h^3)\\ \omega_{n-2,j} &= \omega_{n,j} - 2h\left.\frac{\partial \omega}{\partial \xi}\right|_{(n,j)} + \frac{(2h)^2}{2}\left.\frac{\partial^2 \omega}{\partial \xi^2}\right|_{(n,j)} + \mathrm{O}(h^3).\end{aligned}$$

We assume that the first derivative terms are zero and multiply the first equation by four and subtract the second equation to find

$$4\omega_{n-1,j} - \omega_{n-2,j} = 3\omega_{n,j} + \mathrm{O}(h^3).$$

The boundary condition on $\omega_{n,j}$ is therefore

$$\omega_{n,j} = \frac{4\omega_{n-1,j} - \omega_{n-2,j}}{3}.$$

Solutions to the Problems for Lecture 12

1. Near the cylinder surface, we assume that

$$\psi(\xi) = a\xi^2.$$

Then

$$\left.\frac{\partial^2\psi}{\partial\xi^2}\right|_{(1,j)} = 2a, \qquad \psi_{2,j} = ah^2, \qquad \psi_{3,j} = 4ah^2,$$

and

$$\frac{1}{2h^2}(8\psi_{2,j} - \psi_{3,j}) = \frac{1}{2h^2}(8ah^2 - 4ah^2) = 2a = \left.\frac{\partial^2\psi}{\partial\xi^2}\right|_{(1,j)}.$$

Solutions to the Problems for Lecture 13

1. The SOR iteration formulas are

$$\psi_{i,j}^{k+1} = (1 - r_\psi)\psi_{i,j}^k + \frac{r_\psi}{4}\left(\psi_{i+1,j}^k + \psi_{i-1,j}^k + \psi_{i,j+1}^k + \psi_{i,j-1}^k + h^2 e^{2\xi_i}\omega_{i,j}^k\right),$$
$$\omega_{i,j}^{k+1} = (1 - r_\omega)\omega_{i,j}^k + \frac{r_\omega}{4}\left(\omega_{i+1,j}^k + \omega_{i-1,j}^k + \omega_{i,j+1}^k + \omega_{i,j-1}^k + \frac{\mathrm{Re}}{8}f_{i,j}^k\right),$$

where

$$f_{ij}^k = \left(\psi_{i+1,j}^k - \psi_{i-1,j}^k\right)\left(\omega_{i,j+1}^k - \omega_{i,j-1}^k\right) - \left(\psi_{i,j+1}^k - \psi_{i,j-1}^k\right)\left(\omega_{i+1,j}^k - \omega_{i-1,j}^k\right).$$

The iterations start with ψ and ω everywhere zero except for

$$\psi_{n,j} = e^{\xi_n}\sin\theta_j, \quad \text{for } j = 1 \text{ to } m.$$

Note that every term in the iteration equation for the scalar vorticity ω^{k+1} is proportional to ω^k. Therefore, ω^k can only become nonzero through the boundary condition on the cylinder:

$$\omega_{1,j} = (\psi_{3,j} - 8\psi_{2,j})/2h^2,$$

and ω will first become nonzero after $\psi_{3,j}$ becomes nonzero. In the iteration equation for the stream function $\psi_{i,j}^{k+1}$, there is a term $\psi_{i+1,j}^k$ so that the nonzero boundary condition at $i = n$ propagates downward in i with each iteration until it finally reaches $i = 3$. This will take $n - 3$ iterations of the SOR equations.

Considerations such as this are useful when debugging code and monitoring how the fields develop with each iteration.

Solutions to the Problems for Lecture 14

1. Complete your solution in the MATLAB Grader using the Learner Template:

```
function [psi, omega] = flow_around_cylinder_steady
Re=10;
%%%%% define the grid %%%%%
n=101; m=101; % number of grid points
N=n-1; M=m-1; % number of grid intervals
h=pi/M; % grid spacing based on theta variable
xi=(0:N)*h; theta=(0:M)*h; % xi and theta variables on the grid
%%%%% Initialize the flow fields %%%%%
psi=zeros(n,m);
omega=zeros(n,m);
psi(n,:)=... % Write the free-stream boundary condition here
%%%%% Set relax params, tol, extra variables %%%%%
r_psi=1.8; r_omega=0.9; % relaxation parameters
delta=1.e-08; % error tolerance
error=2*delta; % initialize error variable
%%%%% Add any additional variable definitions here %%%%%
...
...
%%%%% Main SOR Loop %%%%%
while (error > delta)
    psi_old = psi; omega_old = omega;
    for i=2:n-1
        for j=2:m-1
            psi(i,j)=... % Write the psi equation here
        end
    end
    error_psi=max(abs(psi(:)-psi_old(:)));
    omega(1,:)=... % Write the boundary condition here
    for i=2:n-1
        for j=2:m-1
            omega(i,j)=... % Write the omega equation here
        end
    end
    error_omega=max(abs(omega(:)-omega_old(:)));
    error=max(error_psi, error_omega);
end
plot_Re10(psi);
```

Solutions to the Problems for Lecture 15

1. Here is a copy of the graphics code.

```
function  plot_Re10(psi)
Re=10;
%%%%% xi-theta grid %%%%%
n=size(psi,1); m=size(psi,2);
N=n-1; M=m-1;
h=pi/M; % grid spacing
xi=(0:N)*h; theta=(0:M)*h;
[XI, THETA] = meshgrid(xi,theta);
%%%%%% x-y grid %%%%%
nx=640; ny=480/2; % number of pixels in x and half of y
xmin=-1.5; xmax=2.5; ymax=(xmax-xmin)*ny/nx; ymin=-ymax;
x=linspace(xmin,xmax,nx+1);  y=linspace(0,ymax,ny+1);
[X,Y]=meshgrid(x,y);
%%%%% construct interpolation points %%%%%
xi_i=0.5*log(X.^2+Y.^2);
theta_i=wrapTo2Pi(atan2(Y,X));
%%%%% interpolate %%%%%
psi_xy=interp2(XI,THETA,psi',xi_i,theta_i);
%%%%% set psi to zero inside cylinder %%%%%
psi_xy(xi_i<0)=0;
%%%%% scale contour levels %%%%%
%%%%% negative values have same range as positive values %%%%%
psi_min=min(psi_xy(:));
psi_max=max(psi_xy(:));
psi_xy(psi_xy<0)=psi_xy(psi_xy<0)/abs(psi_min);
psi_xy(psi_xy>0)=psi_xy(psi_xy>0)/abs(psi_max);
%%%%% set colormap for contours %%%%%
levels=linspace(-1,1,1000);
cmap=flipud(jet(length(levels)));
colormap(cmap);
%%%%% plot color contours %%%%%
imagesc(x,y,psi_xy); hold on;
imagesc(x,-y,-psi_xy); % negative values of y
%%%%% plot contour lines %%%%%
v=[-0.9:0.2:-0.1,0,0.0005,0.001,0.002:0.004:0.01,0.02:0.04:0.9];
contour(X,Y,psi_xy,v,'LineColor','k');
contour(X,-Y,-psi_xy,-v,'LineColor','k');
%%%%% draw black circle for cylinder %%%%%
t=linspace(0,2*pi, 1000);
a=cos(t); b=sin(t);
fill(a,b,[0 0 0]);
%%%%% beautify plot %%%%%
set(gca,'YDir','normal');
axis([xmin xmax ymin ymax]); set(gcf,'color','w'); axis equal; axis off;
text(xmin+0.75*(xmax-xmin),ymin+0.08*(ymax-ymin),...
                  ['Re = ', num2str(Re,'%3.0f')],'FontSize',22,'Color','k');
```

Solutions to the Problems for Lecture 16

1. Periodic boundary conditions requires for the end points

$$\psi_{i,1} = \psi_{i,m-1}, \quad \omega_{i,1} = \omega_{i,m-1}; \qquad \psi_{i,m} = \psi_{i,2}, \quad \omega_{i,m} = \omega_{i,2}.$$

The boundary conditions are the same as the implementation in the lecture, but the values of θ are now shifted by one grid point, starting at zero rather than $-h$ and ending at $2\pi + h$ rather than at 2π.

Solutions to the Problems for Lecture 17

1. The scalar vorticity equation is given by

$$\frac{\partial \omega}{\partial t} = \frac{2e^{-2\xi}}{\mathrm{Re}} \left(\frac{\partial^2 \omega}{\partial \xi^2} + \frac{\partial^2 \omega}{\partial \theta^2} \right) + e^{-2\xi} \left(\frac{\partial \psi}{\partial \xi} \frac{\partial \omega}{\partial \theta} - \frac{\partial \psi}{\partial \theta} \frac{\partial \omega}{\partial \xi} \right).$$

Second-order finite difference approximations for the first- and second-derivatives yields

$$\frac{d\omega_{i,j}}{dt} = \frac{2e^{-2\xi_i}}{\mathrm{Re}} \left[\left(\frac{\omega_{i+1,j} - 2\omega_{i,j} + \omega_{i-1,j}}{h^2} \right) + \left(\frac{\omega_{i,j+1} - 2\omega_{i,j} + \omega_{i,j-1}}{h^2} \right) \right] + e^{-2\xi_i} \left[\left(\frac{\psi_{i+1,j} - \psi_{i-1,j}}{2h} \right) \left(\frac{\omega_{i,j+1} - \omega_{i,j-1}}{2h} \right) - \left(\frac{\psi_{i,j+1} - \psi_{i,j-1}}{2h} \right) \left(\frac{\omega_{i+1,j} - \omega_{i-1,j}}{2h} \right) \right].$$

Collecting terms yields

$$\frac{d\omega_{i,j}}{dt} = \frac{2e^{-2\xi_i}}{h^2 \mathrm{Re}} \left(\omega_{i+1,j} + \omega_{i-1,j} + \omega_{i,j+1} + \omega_{i,j-1} - 4\omega_{i,j} \right) + \frac{e^{-2\xi_i}}{4h^2} \left[\left(\psi_{i+1,j} - \psi_{i-1,j} \right) \left(\omega_{i,j+1} - \omega_{i,j-1} \right) - \left(\psi_{i,j+1} - \psi_{i,j-1} \right) \left(\omega_{i+1,j} - \omega_{i-1,j} \right) \right].$$

Solutions to the Problems for Lecture 18

1. One straightforward implementation that uses two `for` loops is

```
omega_tilde=zeros(n,m);
for j=1:m
    for i=1:n
        omega_tilde(i,j}=h^2*exp(2*xi(i))*omega(i,j);
    end
end
omega_tilde = omega_tilde(:);
```

An implementation that doesn't use `for` loops could use the MATLAB `repmat.m` function to duplicate `xi` across `m` columns, and may be written as

```
omega_tilde=h^2*repmat(exp(2*xi(:)),1,m).*omega;
omega_tilde=omega_tilde(:);
```

Solutions to the Problems for Lecture 19

1. Only the fifth row satisfies the Poisson equation, being the only internal grid point. Rows one to three and rows seven to nine obey periodic boundary conditions, row four satisfies the cylinder boundary condition $\psi_4 = 0$ and row six satisfies the free-stream condition $\psi_6 = e^{\xi_3} \sin\theta_2$.

$$\begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \\ \Phi_7 \\ \Phi_8 \\ \Phi_9 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ h^2 e^{2\xi_2} \omega_{2,2} \\ e^{\xi_3} \sin\theta_2 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

With only three grid points in θ, all the θ values are actually located at the same point (equivalent to $\theta = 0$)! Nevertheless, this result can be used to debug your construction of the matrix for the stream-function equation.

Solutions to the Problems for Lecture 20

1. We have found that the line of code that solves the stream-function equation $\mathrm{A}\psi = \mathrm{b}$ using the backslash operator takes almost 90% of the computational time. Replacing A by LU, where the LU decomposition need only be done once, saves substantial computational time.

Solutions to the Problems for Lecture 21

1. Complete your solution in the MATLAB Grader using the Learner Template:

```
function omega = flow_around_cylinder_unsteady
Re=60;
%%%%% define the grid %%%%%
n=101; m=202; % number of grid points
N=n-1; M=m-2; % number of grid intervals: 2 ghost points, theta=-h,2*pi
h=2*pi/M; % grid spacing based on theta variable
xi=(0:N)*h; theta=(-1:M)*h; % xi and theta variables on the grid
%%%%% time-stepping parameters %%%%%
t_start=0; t_end=0.5; % vortex street starts at around t=900
tspan=[t_start t_end];
%%%%% Initialize vorticity field %%%%%
omega=zeros(n,m);
%%%%% Construct the matrix A for psi equation %%%%%


%%%%% Find the LU decomposition %%%%%
[L,U]=lu(A); clear A;
%%%%% Compute any time-independent constants %%%%%


%%%%% advance solution using ode23 %%%%%
options=odeset('RelTol', 1.e-03);
omega=omega(2:n-1,2:m-1); % strip boundary values for ode23
omega=omega(:); % make a column vector
[t,omega]=ode23...
  (@(t,omega)omega_eq(omega,L,U, (additional arguments),...
                                                  tspan, omega, options);
%%%%% expand omega to include boundaries %%%%%
temp=zeros(n,m);
temp(2:n-1,2:m-1)=reshape(omega(end,:),n-2,m-2);
omega=temp; clear temp;
%%%%% compute stream function (needed for omega boundary values) %%%%%


%%%%% set omega boundary conditions %%%%%


%%%%% plot scalar vorticity field %%%%%
plot_Re60(omega,t_end);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function d_omega_dt=omega_eq(omega,L,U,%additional arguments)
%%%%% expand omega to include boundary points %%%%%
temp=zeros(n,m);
index1=2:n-1; index2=2:m-1;
temp(index1,index2)=reshape(omega,n-2,m-2);
omega=temp; clear temp;
%%%%% compute stream function %%%%%

%%%%% compute derivatives of omega %%%%%

end
```

Here is a copy of the code `plot_Re60.m`.

```
function  plot_Re60(omega,t_plot)
% Plot vorticity for Re = 60 from flow_past_circle_unsteady
Re=60;
% xi-theta grid
n=size(omega,1); m=size(omega,2);
N=n-1; M=m-2;
h=2*pi/M; %grid spacing
xi=(0:N)*h; theta=(-1:M)*h; %xi and theta variables on the grid
[XI, THETA] = meshgrid(xi,theta);

% x-y grid
nx=640; ny=480; %number of pixels in x and y
xmin=-1.5; xmax=10; ymax=((xmax-xmin)/2)*ny/nx; ymin=-ymax;
x=linspace(xmin,xmax,nx+1);  y=linspace(ymin,ymax,ny+1);
[X,Y]=meshgrid(x,y);

%construct interpolation points
xi_i=0.5*log(X.^2+Y.^2);
theta_i=wrapTo2Pi(atan2(Y,X));

omega_xy=interp2(XI,THETA,omega',xi_i,theta_i);

%inside circle
omega_xy(xi_i<0)=0;

%set colormap for contours
levels=linspace(-1,1,1000);
v=[levels(1) levels(end)];
cmap=jet(length(levels));
cmap=flipud(cmap);
colormap(cmap);

%color contours
imagesc(x,y,omega_xy,v); hold on;

%draw black circle for cylinder
t=linspace(0,2*pi, 1000);
a=cos(t); b=sin(t);
fill(a,b,[0 0 0]);

%neaten plot
set(gca,'YDir','normal');
axis([xmin xmax ymin ymax]); set(gcf,'color','w'); axis equal; axis off;
text(xmin+0.75*(xmax-xmin),ymin+0.08*(ymax-ymin),...
              ['t = ', num2str(t_plot,'%3.1f')],'FontSize',22,'Color','k');
```

Solutions to the Problems for Lecture 22

1. Here is a copy of the graphics code.

```
function  plot_Re60_movie
Re=60;
%%%%% Initialize movie file %%%%%
writerObj = VideoWriter('movie.mp4','MPEG-4');
writerObj.Quality = 100;
writerObj.FrameRate=24;
open(writerObj);
input_file='.\fields\Re_60_';
%%%%% Loop over fields to construct plot and put in movie file %%%%%
nfields=1000;
for ij=1:nfields
    load([input_file num2str(ij)],'omega');
    %%%%% xi-theta grid %%%%%
    n=size(omega,1); m=size(omega,2);
    N=n-1; M=m-2;
    h=2*pi/M; %grid spacing
    xi=(0:N)*h; theta=(-1:M)*h;
    [XI, THETA] = meshgrid(xi,theta);
    %%%%%% x-y grid %%%%%
    nx=640; ny=480; % number of pixels in x and y
    xmin=-1.5; xmax=20; ymax=((xmax-xmin)/2)*ny/nx; ymin=-ymax;
    x=linspace(xmin,xmax,nx+1);  y=linspace(ymin,ymax,ny+1);
    [X,Y]=meshgrid(x,y);
    %%%%% construct interpolation points %%%%%
    xi_i=0.5*log(X.^2+Y.^2);
    theta_i=wrapTo2Pi(atan2(Y,X));
    %%%%% interpolate %%%%%
    omega_xy=interp2(XI,THETA,omega',xi_i,theta_i);
    %%%%% set omega to zero inside cylinder %%%%%
    omega_xy(xi_i<0)=0;
    %%%%% set colormap for contours %%%%%
    levels=linspace(-1,1,1000);
    v=[levels(1) levels(end)];
    cmap=flipud(jet(length(levels)));
    colormap(cmap);
    %%%%% plot color contours %%%%%
    imagesc(x,y,omega_xy,v); hold on;
    %%%%% draw black circle for cylinder %%%%%
    t=linspace(0,2*pi, 1000);
    a=cos(t); b=sin(t);
    fill(a,b,[0 0 0]);
    %%%%% beautify plot %%%%%
    set(gca,'YDir','normal');
    axis([xmin xmax ymin ymax]); set(gcf,'color','w'); axis equal; axis off;
    text(xmin+0.75*(xmax-xmin),ymin+0.08*(ymax-ymin),...
         ['t = ', num2str(ij,'%3.0f')],'FontSize',22,'Color','k');
    %%%%% add frame to movie file %%%%%
    frame=getframe(gcf);
    writeVideo(writerObj,frame);
    close;
    fprintf('%g\n',ij)
end
close(writerObj);
```