The Mathematica® Journal

Airfoil Aerodynamics Using Panel Methods

Richard L. Fearn

Potential flow over an airfoil plays an important historical role in the theory of flight. The governing equation for potential flow is Laplace's equation, a widely studied linear partial differential equation. One of Green's identities can be used to write a solution to Laplace's equation as a boundary integral. Numerical models based on this approach are known as panel methods in the aerodynamics community. This article introduces the availability of a collection of computational tools for constructing numerical models for potential flow over an airfoil based on panel methods. Use of the software is illustrated by implementing a specific model using vortex panels of linearly varying strength to compute the flow over a member of the NACA four-digit family of airfoils.

Introduction

Fluid dynamics is a branch of mechanics concerned with the motion of a fluid continuum under the action of applied forces. The motion and general behavior of a fluid is governed by the fundamental laws of classical mechanics and thermodynamics and plays an important role in such diverse fields as biology, meteorology, chemical engineering, and aerospace engineering. An introductory text on fluid mechanics, such as [1], surveys the basic concepts of fluid dynamics and the various mathematical models used to describe fluid flow under different restrictive assumptions. Advances in computational power and in modeling algorithms during the past few decades have enabled industry to use increasingly realistic models to solve problems of practical geometric complexity. Alternatively, these advances make it feasible to adapt some of the older, simpler models to inexpensive desktop computers.

Aerodynamics is a branch of fluid dynamics concerned primarily with the design of vehicles moving through air. In the not-so-distant past, a collection of relatively simple numerical models, known as panel methods, was the primary computational tool for estimating some of the aerodynamic characteristics of airplanes and their components for cruise conditions. For example, Hess [2] commented in 1990 that at Douglas Aircraft Company, a major design calculation was performed using panel methods approximately 10 times per day.

Panel methods are numerical models based on simplifying assumptions about the physics and properties of the flow of air over an aircraft. The viscosity of air in the flow field is neglected, and the net effect of viscosity on a wing is summarized by requiring that the flow leaves the sharp trailing edge of the wing smoothly. The compressibility of air is neglected, and the curl of the velocity field is assumed to be zero (no vorticity in the flow field). Under these assumptions, the vector velocity describing the flow field can be represented as the gradient of a scalar velocity potential, $\vec{Q} = \nabla\phi$, and the resulting flow is referred to as potential flow. A statement of conservation of mass in the flow field leads to Laplace's equation as the governing equation for the velocity potential, $\nabla^2\phi = 0$. Laplace's equation is a widely studied linear partial differential equation and is discussed in detail in classical books on applied mathematics such as [3]. It also plays an important role in the theoretical development of several fields, including electrostatics and elastic membranes as well as fluid dynamics.

To solve the problem of potential flow over a solid object, Laplace's equation must be solved subject to the boundary condition that there be no flow across the surface of the object. This is usually referred to as the tangent-flow boundary condition. Additionally, the flow far from the object is required to be uniform. The results of solving Laplace's equation subject to tangent-flow boundary conditions provide an approximation of cruise conditions for an airplane.

Using a vector identity, the solution to this linear partial differential equation can be written in terms of an integral over the surface of the object. This boundary integral contains expressions for surface distributions of basic singular solutions to Laplace's equation. A linear combination of relatively simple singular solutions is also a solution to the differential equation. This superposition of simple solutions provides the complexity needed for satisfying boundary conditions for flow over objects of complex geometry. Panel methods are based on this approach and are described in detail in [4]. Commonly used singular solutions for panel methods are referred to as source, vortex, and doublet distributions. Analogies can be made to other fields of study. The velocity field induced by a point source is analogous to the electrostatic field induced by a point charge. A doublet would be positive and negative charges of equal strength in close proximity. The velocity induced by a line vortex is analogous to the magnetic field induced by a current-carrying wire.

The basic solution procedure for panel methods consists of discretizing the surface of the object with flat panels and selecting singularities to be distributed over the panels in a specified manner, but with unknown singularity-strength parameters. Since each singularity is a solution to Laplace's equation, a linear combination of the singular solutions is also a solution. The tangent-flow boundary condition is required to be satisfied at a discrete number of points called collocation points. This process leads to a system of linear algebraic equations to be solved for the unknown singularity-strength parameters. Details of the procedure vary depending on the singularities used and other details of problem formulation, but the end result is always a system of linear algebraic equations to be solved for the unknown singularity-strength parameters.

Panel methods are applicable to two- and three-dimensional flows. For flow over a two-dimensional object, the flat panels become straight lines, but can be thought of as

infinitely long rectangular panels in the three-dimensional interpretation. For two-dimensional potential flow, the powerful technique of conformal mapping can also be used as a solution procedure. Conformal mapping provides exact solutions for certain airfoil shapes and is useful for validating numerical models.

This article introduces a collection of three packages providing computational tools for the formulation and solution of steady potential flow over an airfoil. In addition to the packages and associated online help for functions defined, examples of model implementation and use are included. Each package is discussed briefly. This is followed by an example of step-by-step implementation of a particular model for a small discretization number with intermediate results displayed. Finally, the steps are assembled into a module representing a particular model, and the lift and pressure distribution on an airfoil are computed. The current version of the software collection is available form the Wolfram Library Archive as *Aerodynamics* 1.2.

Load Required Packages

Set your working directory and then load the application package needed for this notebook.

```
Get["CartesianVectors`"]
```

```
Get["AirfoilGeometry`"]
```

```
Get["InfluenceCoefficients`"]
```

```
Get["Graphics`"]
```

Test that packages are loaded.

```
?CartesianVectors
```

```
CartesianVectors is the name of a data type specifying
    a collection of n−dimensional vectors in a rectangular
    coordinate system. CartesianVectors is also the name of the
    package defining the data type. For revision of TMJ notebook.
```

```
? AirfoilGeometry
```

AirfoilGeometry is a package providing functions to specify airfoil shape, discretize the airfoil and compute geometric properties of the discretizated airfoil. For revision of TMJ notebook.

```
? InfluenceCoefficients
```

InfluenceCoefficients is a package containing functions to compute geometric influence coefficients for two−dimensional potential−flow point and line singularities. Functions for velocity, velocity−potential and stream−function influence coefficients are defined for commonly used distributions of source, doublet and vortex singularities. For revision of TMJ notebook.

```
? Graphics
```

Graphics is a package providing functions for plots where conventions typicall used in the aerodynamics community differ from those used in Mathematica. Also included are utility graphics used in the Aerodynamics software. For revision of TMJ notebook. ≫

Nomenclature and Basic Equations for Airfoil Aerodynamics

Much of the nomenclature associated with the theory of lift on an airfoil has made its way into everyday vocabulary, but some terms may be unfamiliar or have more specific meanings than occur in common usage. Also, there are some basic equations used in the example problem that should be mentioned. An introductory book on aerodynamics such as [5] or [6] presents the basic nomenclature and concepts associated with the theory of flight.

The term *airfoil* is used to denote the cross section, or profile, of a three-dimensional wing (see inset in Figure 2). The *chord line* of an airfoil is the straight line from the leading edge of the airfoil to the sharp trailing edge; the length of this line is referred to as the chord of the airfoil and is denoted by c. The *camber line* of an airfoil is the locus of points midway between the upper and lower surfaces of the airfoil, measured perpendicular to the camber line. When describing an airfoil in dimensionless variables and in a local coordinate system, the chord of the airfoil is the segment of the x axis from 0 to 1. The *angle of attack* is the angle between the chord line of the airfoil and the uniform onset velocity, and is denoted by α.

In incompressible potential flow, the pressure is related to the fluid speed by Bernoulli's equation, $\frac{1}{2}\rho Q^2 + p = \frac{1}{2}\rho Q_\infty^2 + p_\infty$, where ρ, Q, and p are the density, speed, and pressure at a point in the flow field, and the subscript ∞ refers to conditions far from the airfoil. The dimensionless measure of pressure is the pressure coefficient, defined by $C_p = \frac{p - p_\infty}{\frac{1}{2}\rho Q_\infty^2}$. Combining Bernoulli's equation and the definition for the pressure coefficient yields a simple equation for the pressure coefficient in terms of the local speed of the fluid, $C_p = 1 - \frac{Q^2}{Q_\infty^2}$.

Using the aerodynamic sign convention, the *circulation* of the velocity field $\vec{Q}$ around a closed contour C is defined by the line integral $\Gamma = -\oint_C \vec{Q} \cdot d\vec{s}$. The lift force per unit length on an airfoil can be related to the circulation around the airfoil by the Kutta–Joukowski lift theorem $\ell = \rho Q_\infty \Gamma$. The aerodynamic sign convention used in the definition of circulation is chosen so that positive circulation leads to positive lift. The dimensionless measure for lift on an airfoil is the two-dimensional lift coefficient, $c_\ell = \frac{\ell}{\frac{1}{2}\rho Q_\infty^2 c}$. If a dimensionless circulation is defined by $\overline{\Gamma} = \frac{\Gamma}{c Q_\infty}$, then the lift coefficient is simply twice the dimensionless circulation. The *Kutta condition* summarizes the primary viscous effect of the flow on the airfoil and establishes the circulation around the airfoil by the simple statement that the flow leaves the sharp trailing edge of the airfoil smoothly.

Input Parameters

The example airfoil for illustrating the implementation of a panel method in this article is a member of the NACA four-digit family of airfoils. Specify the identification number for the airfoil and the angle of attack in degrees.

```
id = 4412; α = 10.0 Degree;
```

The first of the four digits in the identification number gives the maximum camber in percent chord, the second digit gives the location of maximum camber in tenths of chord, and the last two digits give the thickness in percent chord.

The discretization process is determined by a discretization number and one of three layout options (`ConstantSpacing`, `CosineSpacing`, or `HalfCosineSpacing`) providing two alternatives to constant spacing of discretization points. Specify small (to illustrate a step-by-step implementation of the example) and large (for computing results) discretization numbers, and a layout option.

```
ns = 3; nl = 100; spacing = HalfCosineSpacing;
```

Finally, specify a small number used to ensure that collocation points are computed to be outside of the discretization panels representing the airfoil.

```
ϵ = 10.^-6;
```

If you have installed the software discussed in this article, you can change the input parameters and rerun the notebook for additional results.

Packages

Data Type for Handling Collections of Vectors

The package *CartesianVectors* defines a data type to simplify the manipulation of large collections of *n*-dimensional vectors while maintaining packed arrays for efficient computation using machine numbers. The data type `CartesianVectors` is represented in the format `{vx↑vy↑vz}`, where `vx`, `vy`, and `vz` are simple or nested lists of the components of the collection of vectors. The data type is designed to enable the manipulation of collections of vectors with notation commonly used for a single vector or for lists. Using a data type also simplifies pattern matching for valid input arguments for exported functions developed in other packages. The properties of the data type are defined by overloading existing *Mathematica* functions whenever possible. The package contains some exported functions including several functions specific to two-dimensional vectors.

As an example of using the data type, specify two collections of two-dimensional vectors, and compute the collection of displacement vectors from each vector in one group to every vector in the other group. This is a computation common to many *n*-body problems. The constructor for the data type is `MakeCartesianVectors`.

```
rA = MakeCartesianVectors[{Array[xa, 3], Array[za, 3]}]
```

```
{xa[1], xa[2], xa[3]}↑{za[1], za[2], za[3]}
```

```
rB = MakeCartesianVectors[{Array[xb, 2], Array[zb, 2]}]
```

```
{xb[1], xb[2]}↑{zb[1], zb[2]}
```

Compute the displacement vectors from each point in `rB` to all points in `rA`.

```
(rAB = Outer[Plus, rA, -rB]) // MatrixForm
```

$$\begin{pmatrix} xa[1]-xb[1] & xa[1]-xb[2] \\ xa[2]-xb[1] & xa[2]-xb[2] \\ xa[3]-xb[1] & xa[3]-xb[2] \end{pmatrix} \uparrow \begin{pmatrix} za[1]-zb[1] & za[1]-zb[2] \\ za[2]-zb[1] & za[2]-zb[2] \\ za[3]-zb[1] & za[3]-zb[2] \end{pmatrix}$$

Count the number of vectors in the collection of displacement vectors.

```
NumberOfVectors[rAB]
```

```
6
```

□ Airfoil Geometry

The package *AirfoilGeometry* provides functions to compute the geometry and discretization of airfoils in support of the construction of numerical models for potential flow over an airfoil. A list of x values used for discretization can be specified directly by the user or generated by the function `NDiscretizeUnitSegment`, which accepts a discretization number, n, as its input argument and divides the unit segment into n pieces. A layout option for this function allows constant spacing (default), cosine spacing, or half-cosine spacing. Cosine spacing provides finer discretization near the leading and trailing edges of the airfoil compared to constant spacing, and half-cosine spacing provides even finer discretization near the leading edge, but coarser discretization near the trailing edge compared to constant spacing. The function `NACA4DigitAirfoil` computes a list of thickness and camber properties at the x values, and the function `AirfoilSurfacePoints` computes the collection of vectors locating points on the surface of the airfoil from the list of thickness and camber properties. These points on the surface of the airfoil serve as panel end points for the discretized airfoil. Note that the result is expressed as the data type `CartesianVectors` as indicated by the arrow separating the lists of components.

```
rPanels = AirfoilSurfacePoints[
  NACA4DigitAirfoil[id,
   NDiscretizeUnitSegment[ns, Layout → spacing]]]
```

```
{0.999833, 0.498824, 0.140789, 0., 0.127161,
  0.501176, 1.00017}↑{-0.00124895, -0.0140383,
  -0.0289205, 0., 0.0735357, 0.0918161, 0.00124895}
```

Compute a list of panel lengths. Note the use of *Mathematica* functions that have been overloaded for use with the data type `CartesianVectors`.

```
lengthPanels = PanelLengths[rPanels]
```

```
{0.501173, 0.358344, 0.143728, 0.146892, 0.374462, 0.507143}
```

Count the number of panels describing the discretized airfoil.

```
nPanels = Length[lengthPanels]
```

```
6
```

The functions `PanelPoints` and `PanelNormals` are used to locate collocation points at midpanel and outward facing unit normals to the panels. Note that collocation points are displaced a small distance, proportional to the panel length, in the direction of the outward unit normal to ensure that these points are outside the discretized airfoil. This is done in preparation for applying the tangent-flow boundary condition at collocation points.

```
unPanels = PanelNormals[rPanels]
```

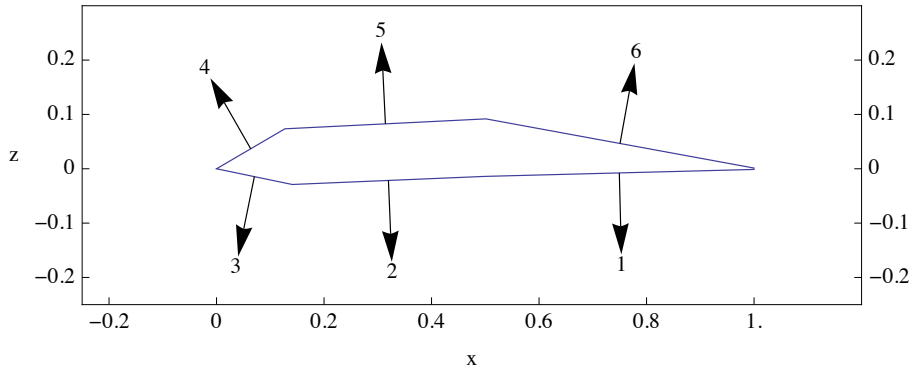
```
{0.0255188, 0.0415304, -0.201216, -0.50061,
  -0.0488176, 0.178583}↑{-0.999674, -0.999137,
  -0.979547, 0.865673, 0.998808, 0.983925}
```

```
rCollocation = PanelPoints[rPanels] +
  ε MultiplyByList[lengthPanels, unPanels]
```

```
{0.749329, 0.319806, 0.0703943, 0.0635802,
  0.314168, 0.750671}↑{-0.00764412, -0.0214798,
  -0.0144604, 0.036768, 0.0826763, 0.046533}
```

Figure 1 shows the geometry of the discretized airfoil and the numbering convention for panels. The panels are straight-line segments joining points on the airfoil contour, and panel normals are shown at panel midpoints with the panel number near the head of the arrow. For an airfoil with thickness, the number of panels describing the airfoil is twice the discretization number. This numbering of panels is referred to as the clockwise convention. For a reference airfoil with no thickness (camber line), the number of panels is equal to the discretization number, and the convention is to number panels from leading edge to trailing edge. The airfoil shape is plotted in a local coordinate system with the origin at the leading edge of the airfoil and the x axis coincident with the chord line. Lengths are nondimensionalized using the chord of the airfoil, c.

```
gPanels = PlotAirfoil[rPanels];
gNormals =
  Graphics[Arrow[rCollocation,
    rCollocation + 0.15 unPanels]];
Show[gPanels, gNormals,
 Epilog →
  {Table[Text[i, rCollocation[[i]] + 0.17 unPanels[[i]]],
    {i, NumberOfVectors[rCollocation]}]},
 PlotRange → {{-0.25, 1.2}, {-0.25, 0.3}}]
```



▲ **Figure 1.** Panels and panel normals for NACA 4412 airfoil discretized to six panels.

The online help for this package also includes examples of importing data files for individual airfoils. These are defined by specifying points on the airfoil contour and rearranging the imported data for use with this software. The UIUC Airfoil Data Site, maintained by Michael Selig of the University of Illinois at Urbana-Champaign, contains specifications for over 1500 airfoils [7].

□ Two-Dimensional Influence Coefficients

When computing the velocity field induced at a point due to a singularity located elsewhere, the velocity can be written as the product of a geometric term (called an influence coefficient) and a measure of the strength of the singularity. For example, consider the velocity induced at an arbitrary field point r_f due to a point source located at the origin of the coordinate system.

```
rf = {x, z}
```

```
{x, z}
```

```
rs = {0, 0}
```

```
{0, 0}
```

Compute the velocity, velocity-potential, and stream-function influence coefficients at the field point due to the singularity using the package functions `ICSourcePoint`, `ICPhiSourcePoint`, and `ICPsiSourcePoint`.

```
{ic, icϕ, icψ} = {ICSourcePoint[rf, rs],
  ICPhiSourcePoint[rf, rs], ICPsiSourcePoint[rf, rs]}
```

$$\left\{\left\{\frac{x}{2\,\pi\,\left(x^2+z^2\right)},\ \frac{z}{2\,\pi\,\left(x^2+z^2\right)}\right\},\ \frac{\text{Log}\left[x^2+z^2\right]}{4\,\pi},\ \frac{\text{ArcTan}[x,\ z]}{2\,\pi}\right\}$$

The velocity, velocity-potential, or stream-function at r_f would be obtained by multiplying the appropriate influence coefficient by the strength of the source. Influence coefficients can also be thought of as the velocity, velocity-potential, or stream-function induced by a singularity of unit strength.

The package *InfluenceCoefficients* contains over thirty functions for velocity, velocity-potential, and stream-function influence coefficients for source, vortex, and doublet singularities commonly used in two-dimensional panel methods. They serve as a tool box for constructing numerical models for two-dimensional potential flow.

■ Potential-Flow Model Using Vortex Panels of Linearly Varying Strength

□ Step-by-Step Model Formulation Using Coarse Discretization

The singularity element chosen for this model is the vortex panel of linearly varying strength, which provides a circulation density along the i^{th} panel of the form, $\gamma_i(x_i) = \gamma_{0\,i} + s_i\,x_i$ in local coordinates, where x_i is the distance from the "beginning" of the panel. Each singularity panel involves two unknown constants, $\gamma_{0\,i}$ and s_i.

The boundary condition that the velocity be everywhere tangent to the airfoil contour is discretized to require that the velocity component normal to each panel at the collocation point be zero. Since each vortex panel introduces two unknown strength parameters, application of the tangent-flow boundary condition provides n_{panels} equations and $2\,n_{\text{panels}}$ unknowns, where n_{panels} is the number of panels describing the geometry of the discretized airfoil. Continuity of circulation density from one panel to the next and the Kutta condition provide n_{panels} additional equations to complete a system of $2\,n_{\text{panels}}$ linear algebraic equations and $2\,n_{\text{panels}}$ unknowns. The system of equations can be put into standard form. The terms involving unknowns are collected on the left-hand side of the system of equations and the known quantities are collected on the right-hand side. The result can be written in block-matrix form as

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} \gamma_0 \\ s \end{pmatrix} = \begin{pmatrix} -Q_n \\ 0 \end{pmatrix}.$$

The symbols γ_0 and s represent lists of the unknown constant and linear strength parameters for the vortex panels: a_{11} represents the projection of the panel influence coefficients associated with γ_0 on the unit normal vectors, a_{12} represents the projection of the panel influence coefficients associated with s on the unit normal vectors, a_{21}Math and a_{22} represent terms in the equations imposing continuity of circulation density between panels and the Kutta condition, and Q_n is the projection of the free-stream velocity on unit normals at collocation points.

Use the block-matrix form to write the system of equations as $a_{11}\,\gamma_0 + a_{12}\,s = -Q_n$ and $a_{21}\,\gamma_0 + a_{22}\,s = 0$. Solve the latter system for the list of slope strengths, $s = -a_{22}^{-1}\,a_{21}\,\gamma_0$. Substitute this into the former system of equations to eliminate the slope-strength parameters. The resulting system of equations can be written as $\left(a_{11} - a_{12}\,a_{22}^{-1}\,a_{21}\right)\gamma_0 = -Q_n$. This system of equations can be solved for the list of strength parameters γ_0, and then the transformation is used to compute the list of slope parameters s. All variables in the following formulation and solution are dimensionless.

Compute the matrix of velocity influence coefficients and project them on the panel normals.

```
{ic0, ics} = ICVortexLinear[rCollocation, rPanels];
a11 = ic0.unPanels;
a12 = ics.unPanels;
```

Write the equations expressing the continuity of circulation density between panels and the Kutta condition. The equation expressing the Kutta condition is written to accommodate the different numbering conventions for airfoils with thickness and reference airfoils without thickness.

```
a21 = If[nPanels == 2 ns,
   Module[{d}, d = DiagonalMatrix[Table[1.0, {nPanels}]];
    ReplacePart[-d + RotateLeft[d, {0, 1}], 1., {1, 1}]],
   Module[{d}, d = DiagonalMatrix[Table[1.0, {nPanels}]];
    ReplacePart[-d + RotateLeft[d, {0, 1}], 0, {1, 1}]]];

a22 = RotateRight[DiagonalMatrix[lengthPanels]];
```

Form the coefficient matrix for the system of linear algebraic equations to be solved for unknown strength parameters.

```
a = ArrayFlatten[{{a11, a12}, {a21, a22}}];
```

Display the matrix in reduced precision to illustrate the coefficient matrix for the full system of equations.

```
Chop[NumberForm[MatrixForm[a], {3, 2},
  NumberPadding → {"0", "0"}, NumberSigns → {"-", "+"},
  SignPadding → True]]
```

+0.00	+0.14	+0.03	+0.03	+0.14	-0.08	+0.08	+0.02	+0.00	+0.00	+0.03	-0.08
-0.21	+0.00	+0.09	+0.09	+0.00	-0.20	-0.06	+0.06	+0.01	+0.01	-0.02	-0.04
-0.12	-0.28	+0.00	+0.16	-0.19	-0.12	-0.03	-0.07	+0.02	+0.00	-0.03	-0.03
+0.10	+0.16	-0.16	+0.00	+0.28	+0.11	+0.03	+0.03	-0.02	+0.02	+0.04	+0.02
+0.19	-0.00	-0.08	-0.09	+0.00	+0.21	+0.06	-0.02	-0.01	-0.01	+0.06	+0.04
+0.08	-0.13	-0.03	-0.03	-0.14	+0.00	-0.03	-0.02	-0.00	-0.00	-0.03	+0.08
+1.00	+0.00	+0.00	+0.00	+0.00	+1.00	+0.00	+0.00	+0.00	+0.00	+0.00	+0.51
+1.00	-1.00	+0.00	+0.00	+0.00	+0.00	+0.50	+0.00	+0.00	+0.00	+0.00	+0.00
+0.00	+1.00	-1.00	+0.00	+0.00	+0.00	+0.00	+0.36	+0.00	+0.00	+0.00	+0.00
+0.00	+0.00	+1.00	-1.00	+0.00	+0.00	+0.00	+0.00	+0.14	+0.00	+0.00	+0.00
+0.00	+0.00	+0.00	+1.00	-1.00	+0.00	+0.00	+0.00	+0.00	+0.15	+0.00	+0.00
+0.00	+0.00	+0.00	+0.00	+1.00	-1.00	+0.00	+0.00	+0.00	+0.00	+0.37	+0.00

The upper half of the matrix represents normal-component influence coefficients. The first row of the lower half of the matrix represents terms in an equation implementing the Kutta condition and sums the circulation density at the beginning of the first panel and the circulation density at the end of the last panel. Setting this sum to zero imposes zero circulation at the trailing edge of the airfoil. The remaining rows in the lower half of the matrix are coefficients of terms in the equations requiring that the circulation density at the end of one panel be equal to the circulation density at the beginning of the next panel, $\gamma_{0_j} + \delta_j s_j = \gamma_{0_{j+1}}$, where δ_j denotes the length of the j^{th} panel.

Define the transformation matrix to compute the list of slope parameters (s) from the list of constant parameters (γ_0).

```
sFromγ0 = -Inverse[a22].a21;
```

Compute the free-stream velocity at collocation points.

```
qInf = UniformFlow[nPanels, α];
```

Compute the components of the uniform flow normal to panels at collocation points.

```
qnInf = qInf.unPanels;
```

Solve the system of equations for the list of constant parameters.

```
γ0 = LinearSolve[a11 + a12.sFromγ0, -qnInf]
```

```
{-1.26787, -0.814616, -0.685836, 1.19696, 1.76145, 1.41828}
```

Use the transformation matrix to compute the list of slope parameters.

```
s = sFromγ0.γ0
```

```
{0.90439, 0.359375, 13.0997, 3.8429, -0.91643, -0.296589}
```

The lift coefficient for the airfoil can be computed using the Kutta–Joukowski theorem. Recall that the distribution of circulation on a panel in local panel coordinates can be written as $\gamma_i(x_i) = \gamma_{0\,i} + s_i\, x_i$, where x_i denotes the distance from the leading edge of the panel. The contribution of each panel to the lift is computed and the results summed over all panels.

In terms of the dimensionless variables used in this example, the contribution by each panel to the lift coefficient is just twice the net circulation associated with the panel, which is obtained by integrating the linear circulation density function, $\Delta c_{l\,i} = 2 \int_0^{\delta_i} \gamma_i(x_i)\, d x_i = 2\, \gamma_{0\,i}\, \delta_i + s_i\, \delta_i^2$. Compute contributions of each panel to the airfoil lift coefficient.

```
{ΔclC, ΔclL} = {2.0 γ0 lengthPanels, s lengthPanels^2}
```

```
{{-1.27085, -0.583826, -0.197148,
  0.351648, 1.31919, 1.43855}, {0.227159, 0.0461477,
  0.270611, 0.0829194, -0.128503, -0.0762807}}
```

Sum the two terms for each panel to obtain the list of contributions of each panel to the lift coefficient.

```
Δcl = ΔclC + ΔclL
```

```
{-1.04369, -0.537678, 0.0734631, 0.434568, 1.19069, 1.36226}
```

Sum the panel contributions to obtain the airfoil lift coefficient.

```
cl = Total[Δcl]
```

```
1.47962
```

The computations in this section illustrate the process of model implementation using a coarse discretization so that intermediate results can be viewed; however, the discretization is too coarse to provide useful results.

Remove names from computer memory, except those with values needed in the subsequent section, which presents an example computation of the pressure distribution and lift coefficient for a specified airfoil using a larger discretization number.

```
Apply[Remove, Complement[Names["Global`*"],
  {"id", "nl", "spacing", "α", "ϵ"}]]
```

□ Numerical Model for Fine Discretization

In the following expression, the individual steps for implementing the model in the previous section are collected into a module. Most names for variables have been shortened for conciseness, but should be recognizable.

```
{γ0, s} =
  Module[{a, a11, a12, a21, a22, tsγ, qInf, qnInf, γ, δ},
   rP = AirfoilSurfacePoints[
     NACA4DigitAirfoil[id,
      NDiscretizeUnitSegment[nl, Layout → spacing]]];
   δ = Drop[rP - RotateRight[rP], 1]; lp = √(δ.δ) ;
   np = Length[lp]; un = PanelNormals[rP];
   rC = PanelPoints[rP] + ϵ MultiplyByList[lp, un];
   {ic0, ics} = ICVortexLinear[rC, rP]; a11 = ic0.un;
   a12 = ics.un;
   a21 = If[np == 2 nl,
     Module[{d}, d = DiagonalMatrix[Table[1.0, {np}]];
      ReplacePart[-d + RotateLeft[d, {0, 1}], 1., {1, 1}]],
     Module[{d}, d = DiagonalMatrix[Table[1.0, {np}]];
      ReplacePart[-d + RotateLeft[d, {0, 1}], 0, {1, 1}]]];
   a22 = RotateRight[DiagonalMatrix[lp]];
   tsγ = -Inverse[a22].a21; a = a11 + a12.tsγ;
   qInf = UniformFlow[np, α]; qnInf = qInf.un;
   γ = LinearSolve[a, -qnInf]; {γ, tsγ.γ}];
```

The results of this computation are the singularity strength parameters for all panels.

This model implementation has been validated by computing the results for a van de Vooren airfoil for which an exact solution is known by the method of conformal mapping. Also, convergence and timing studies have been performed and are available as online help documents in the software collection.

□ Pressure and Lift Coefficients

Use previously computed influence coefficients to determine the pressure coefficient at collocation points using Bernoulli's equation.

```
cp = Module[{q, qInf}, qInf = UniformFlow[np, α];
    q = qInf + ic0.γ0 + ics.s; 1 - q.q];
```

Lift and pitching moments can be computed from the pressure distribution. For example, the lift coefficient is computed by approximating the integral, $c_l = -\oint C_p \hat{n} \cdot \hat{\ell}\, ds$, where the integral is over the airfoil contour, C_p is the pressure coefficient, $\hat{n}$ is the outward unit normal to the airfoil surface, and $\hat{\ell}$ is a unit vector perpendicular to the free-stream velocity in the direction of positive lift. The integral is approximated by considering the pressure coefficient constant over each panel, computing the contribution to lift of each panel, and summing the results.

```
clFromCp = Module[{ul},
  ul = MakeCartesianVectors[
    {-Table[Sin[α], {np}], Table[Cos[α], {np}]}];
  ΔclP = -cp (ul.un) lp;
  Total[ΔclP]]
```

```
1.70321
```

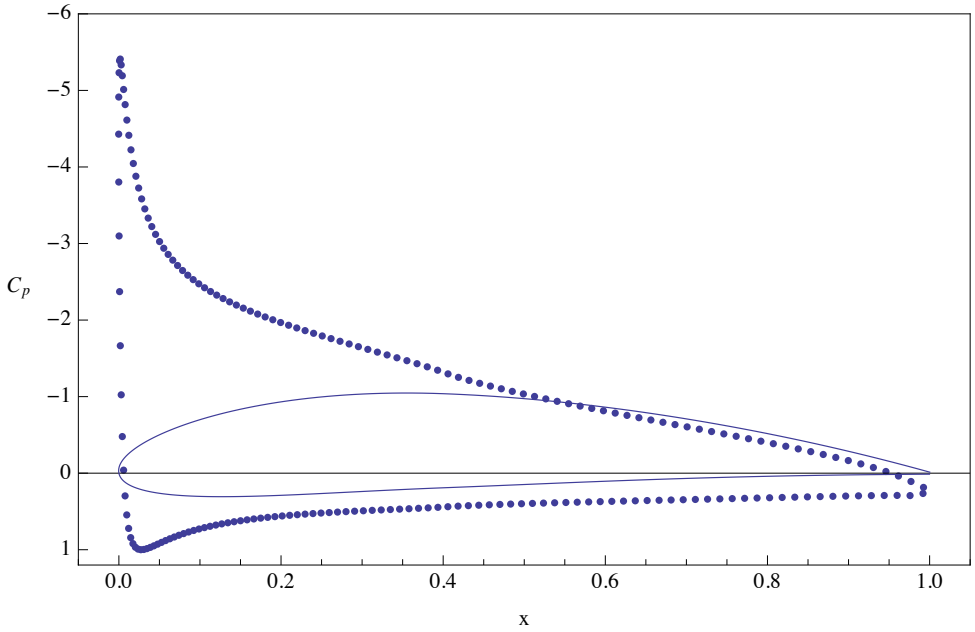
The lift can also be computed from the circulation distribution as described in the section on step-by-step model formulation.

```
cl = Module[{Δcl, ΔclC, ΔclL}, ΔclC = 2.0 γ0 lp; ΔclL = s lp^2;
  Δcl = ΔclC + ΔclL; Total[Δcl]]
```

```
1.71006
```

Figure 2 shows the surface pressure distribution on the airfoil in the conventional manner for such plots. Useful information from such plots include the locations of the stagnation point and the point of minimum pressure, and the severity of the positive pressure gradient on the upper surface.

```
xC = Components[rC][[1]];
PlotSurfacePressureCoefficient[xC, cp,
 Epilog →
  Inset[PlotAirfoil[rP, Frame → False,
    PlotRange → {{0, 1}, Automatic}], {0, 0}, {0, 0}, 1]]
```



▲ **Figure 2.** Pressure coefficient for a NACA 4412 airfoil, $\alpha = 10.0^{\circ}$ discretization 200 panels with `HalfCosineSpacing`.

■ Conclusions

A brief summary of some features of a collection of packages that provide computational tools for formulating numerical models for two-dimensional potential flow over an airfoil using panel methods is presented. An example of solving the problem of steady flow over a specific airfoil is given using vortex panels of linearly varying strength and tangent-flow boundary conditions. This example includes the computation of surface pressure distribution and lift coefficient.

Session time for a typical PC indicates the practicality of such computations on low-cost computing systems and suggests the feasibility of going to the next level of modeling. This could include unsteady two-dimensional potential flow, steady three-dimensional potential flow, or including an integral boundary-layer method with the steady two-dimensional potential flow model presented in this article.

References

[1] R. H. Sabersky, A. J. Acosta, E. G. Hauptmann, and E. M. Gates, *Fluid Flow: A First Course in Fluid Mechanics*, 4th ed., Englewood Cliffs, NJ: Prentice Hall, 1998.

[2] J. L. Hess, "Panel Methods in Computational Fluid Dynamics," *Annual Review of Fluid Mechanics*, **22**, 1990 pp. 255–274.

[3] P. M. Morse and H. Feshbach, *Methods of Theoretical Physics*, New York: McGraw-Hill, 1953.

[4] J. Katz and A. Plotkin, *Low-Speed Aerodynamics*, Cambridge Aerospace Series (No. 13), 2nd ed., New York: Cambridge University Press, 2001.

[5] J. D. Anderson, *Introduction to Flight*, 3rd ed., New York: McGraw-Hill, 1989.

[6] J. J. Bertin and M. L. Smith, *Aerodynamics for Engineers*, 3rd ed., Englewood Cliffs, NJ: Prentice Hall, 1998.

[7] M. S. Selig, "UIUC Airfoil Data Site, Department of Aerospace Engineering." Urbana, Illinois: University of Illinois, (Jan 2007) www.ae.uiuc.edu/m-selig/ads.html.

R. L. Fearn, "Airfoil Aerodynamics Using Panel Methods," *The Mathematica Journal*, 2011. dx.doi.org/10.3888/tmj.10.4-6.

Additional Material

Fearn.zip

Available at library.wolfram.com/infocenter/MathSource/7785/.

About the Author

While teaching for thirty years at the University of Florida, I often wished for effective computational tools to help students learn aerodynamics. Since retirement, I have started developing software for that purpose. When not playing with *Mathematica*, I can usually be found summers hiking in the Canadian Rockies or, spring and fall, walking or canoeing in the northern part of Florida with family and friends.

Richard L. Fearn
Professor Emeritus
Department of Mechanical and Aerospace Engineering
University of Florida
Gainesville, FL 32611-6250
rlf@ufl.edu