A Discrete Vortex Method for Calculating the Minimum Induced Drag and Optimum Load Distribution for Aircraft Configurations with Noncoplanar Surfaces

by
Joel Grasmeyer
Graduate Research Assistant
January, 1997

VPI-AOE-242

W.H. Mason, Faculty Advisor

Multidisciplinary Analysis and Design Center for Advanced Vehicles
Department of Aerospace and Ocean Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061

Please address any questions or bugs to *grasmeye@aoe.vt.edu*

Table of Contents

Introduction

Noncoplanar aircraft configurations can achieve several structural and aerodynamic advantages over coplanar, cantilever configurations. One of the key advantages is the reduction of the induced drag by an increase in the span efficiency. Several noncoplanar configurations have been studied, including biplanes, box planes, ring wings, truss-braced wings, joined wings, C-wings, wings with winglets, and wings with pfeather tips.*

Most of these studies used a discrete vortex method with a Trefftz plane analysis to determine the load distribution corresponding to the minimum induced drag of the configuration. The four main developers of this method are Blackwell[1], Lamar[2], Kuhlman[3], and Kroo[4]. In order to provide this capability at Virginia Tech, a code was implemented in Fortran 77, and several test cases were run to validate the results.

Nomenclature

Theory	FORTRAN	Definition
A	a	influence coefficient matrix (Blackwell)
$\bar{A}$	abar	augmented influence coefficient matrix (Lamar)
AR	ar	aspect ratio of reference surface
$\frac{c_n c}{c_{avg}}$	b	vector of constraints; overwritten with load vector
N/A	bpanel	span of panel (projected onto *xy* plane)
N/A	bref	reference span (projected onto *xy* plane)
c	c	local chord of lifting element
c_{avg}	cavg	average chord of reference surface(s)
C_{D_i}	cd_induced	induced drag coefficient
N/A	cl_actual	actual lift coefficient (obtained via integration)
$C_{L_{design}}$	cl_design	design lift coefficient
N/A	cm_actual	actual pitching moment coefficient (about the *cg*)
$C_{M_{design}}$	cm_design	design pitching moment coefficient (about the *cg*)
N/A	cm_flag	flag for C_m constraint (0 = no constraint, 1 = constraint)
c_n	cn	vector of normal force coefficients
C_P	cp	percent chord location of center of pressure for all sections
N/A	croot	root chord of panel
N/A	ctip	tip chord of panel
N/A	d	dummy output from LU decomposition subroutine
N/A	dist	vortex locations for analysis mode

* The odd spelling honors Werner Pfenninger, one of the first researchers to study the placement of feather tips on wings.

e	e	span efficiency factor
N/A	header	header which identifies the input and output files
i	i	index
N/A	indx	argument used within LU decomposition subroutines
N/A	infile	input filename
N/A	input_mode	input mode (0 = design; find loads, 1 = analysis; loads given)
j	j	index
N/A	k	index
N/A	loads	vector of loads for analysis mode
N/A	load_flag	load flag (0 = cn input, 1 = load (cn*c/cavg) input)
N/A	load_input	input loads specified at given spanwise stations
N/A	load_station	percent span locations of specified loads in the range [0,1]
N/A	n	number of linear equations to be solved
N/A	nconstraints	number of constraints (parameter)
N/A	nloads	number of loads specified per panel for analysis mode input
N/A	np	physical dimension of a array
N/A	npanels	number of panels in geometry
N/A	npanels_max	maximum number of panels in geometry (parameter)
N/A	nvortices	number of vortices per panel
N/A	nvortices_max	maximum number of vortices per panel (parameter)
m	nvortices_tot	total number of vortices
N/A	outfile	output filename
N/A	p	parameter used for linear interpolation
R_1, R_2	r1, r2	vortex influence radii squared
s	s	vortex semi-width (non-dimensional)
N/A	semi_width	temporary parameter for calculating vortex semi-width
N/A	sp	vortex semi-width (dimensional)
N/A	spacing_flag	spacing flag (0 = equal, 1 = outboard-compressed, 2 = inboard-compressed, 3 = end-compressed)
N/A	sref	reference area (projected onto *xy* plane)
N/A	sym_flag	symmetry flag (0 = asymmetric, 1 = symmetric)
N/A	temp_load	temporary load vector for analysis mode
θ	theta	vortex dihedral angle
N/A	title	title of aircraft configuration

N/A	write_flag	write flag (0 = no output file, 1 = output file written)
x, *y*, *z*	x, y, z	coordinates of vortices (aircraft reference frame)
N/A	xc, yc, zc	coordinates of corner points (aircraft reference frame)
x_{cg}	xcg	x location of the center of gravity
x_{le}, x_{te}	xle, xte	coordinates of LE and TE corresponding to each vortex
N/A	yp, zp	coordinates of vortices (vortex reference frame)

Discrete Vortex Algorithm

Coordinate Systems

Figure 1, from Blackwell[1], shows the two coordinate systems used in the discrete vortex method. The first is the traditional aerodynamic reference frame, with the *x*-axis pointing to the rear of the aircraft along the centerline, the *y*-axis pointing starboard, and the *z*-axis pointing up. The second is a set of local reference frames for each vortex element.

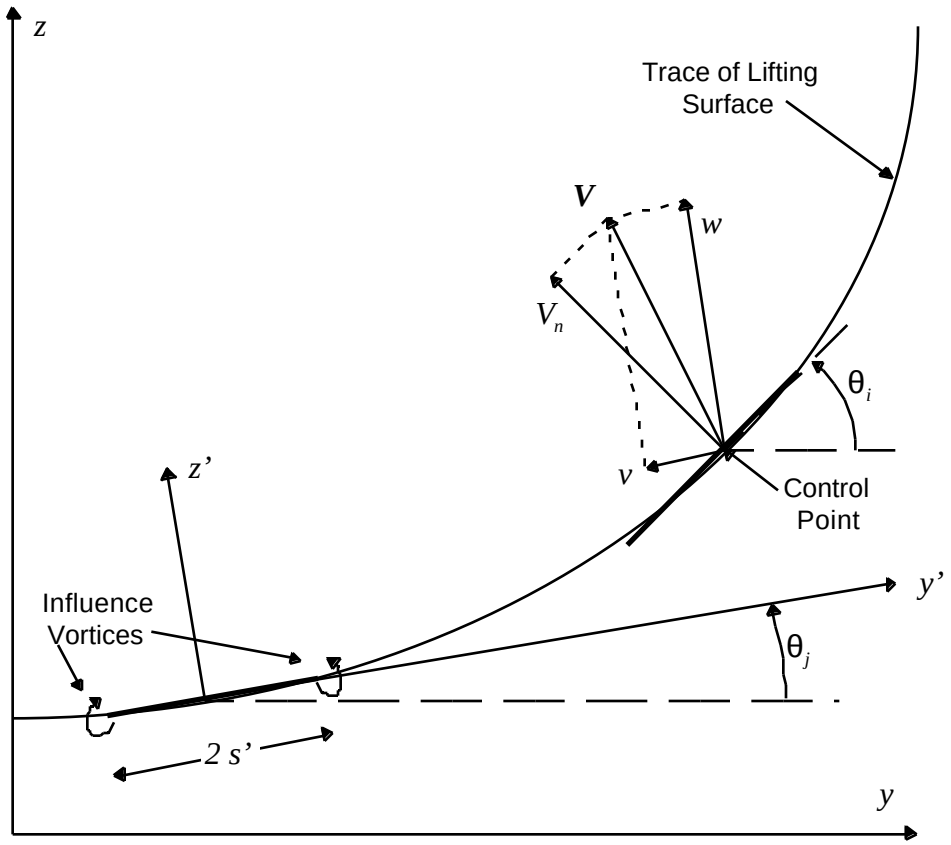

Figure 1: Discrete vortex method coordinate systems (Ref. 1)

Theoretical Development

The discrete vortex method is based on the Kutta-Joukowski theorem, and the Biot-Savart law. These simple expressions, along with some expressions for the configuration geometry can be used to capture the fundamental properties of the flow. The calculations are performed in the Trefftz Plane, which is a vertical plane located far downstream of the aircraft.

In the geometry definition (described later), each aerodynamic surface is represented by a set of discrete horseshoe vortices. The induced drag is then calculated in the Trefftz plane as a function of the velocity induced by the trailing segments of the horseshoe vortices. The Trefftz plane is a plane located at an infinite distance downstream of the aircraft which is perpendicular to the wake. By utilizing the Trefftz plane, the induced drag calculations are independent of the *x*-coordinate, which effectively reduces the 3-dimensional problem to a set of 2-dimensional equations. According to the Biot-Savart law, the velocity induced at a control point P(∞, y_i, z_i) by a horseshoe vortex at P(x_j, y_j, z_j) is given by[1]:

$$\frac{u_i}{V_\infty} = 0 \tag{1}$$

$$\frac{v_i}{V_\infty} = -\frac{1}{2\pi}\frac{\Gamma_j}{V_\infty}\left(\frac{z'}{R_1} - \frac{z'}{R_2}\right) \tag{2}$$

$$\frac{w_i}{V_\infty} = \frac{1}{2\pi}\frac{\Gamma_j}{V_\infty}\left(\frac{(y' - s')}{R_1} - \frac{(y' + s')}{R_2}\right) \tag{3}$$

where:

$$R_1 = (z')^2 + (y' - s')^2 \tag{4}$$

$$R_2 = (z')^2 + (y' + s')^2 \tag{5}$$

$$y' = (y_i - y_j)\cos\theta_j + (z_i - z_j)\sin\theta_j \tag{6}$$

$$z' = -(y_i - y_j)\sin\theta_j + (z_i - z_j)\cos\theta_j \tag{7}$$

According to the Kutta-Joukowski theorem, the circulation is given by[1]:

$$\frac{\Gamma_j}{V_\infty} = \frac{(c_n c)_j}{2} \tag{8}$$

The induced drag coefficient is given by[1]:

$$C_{D_i} = \sum_{i=1}^{m} \frac{V_{n_i}}{V_\infty}\frac{(c_n c)_i}{c_{avg}} s_i \tag{9}$$

where:

$$\frac{V_{n_i}}{V_\infty} = \sum_{j=1}^{m} \frac{(c_n c)_j}{c_{avg}} \left\{ \frac{c_{avg}}{4\pi} \left[\frac{(y' - s')}{R_1} - \frac{(y' + s')}{R_2} \right] \cos(\theta_i - \theta_j) + \frac{c_{avg}}{4\pi} \left[\frac{z'}{R_1} - \frac{z'}{R_2} \right] \sin(\theta_i - \theta_j) \right\} \quad (10)$$

The quantity within the outermost brackets is denoted by A_{ij}, such that[1]:

$$\frac{V_{n_i}}{V_\infty} = \sum_{j=1}^{m} \frac{(c_n c)_j}{c_{avg}} A_{ij} \quad (11)$$

The matrix A_{ij} is known as the "influence coefficient matrix," and it is solely a function of the aircraft geometry. The simplified expression for the induced drag is then obtained by substituting equation (11) into equation (9):

$$C_{D_i} = \sum_{i=1}^{m} \sum_{j=1}^{m} \frac{(c_n c)_j}{c_{avg}} \frac{(c_n c)_i}{c_{avg}} s_i A_{ij} \quad (12)$$

The lift coefficient is given by[1]:

$$C_L = \sum_{j=1}^{m} \frac{(c_n c)_j}{c_{avg}} s_j \cos\theta_j \quad (13)$$

Similarly, the moment coefficient is given by:

$$C_M = \sum_{j=1}^{m} \frac{(c_n c)_j}{c_{avg}} s_j \cos\theta_j \frac{\left[x_{cg} - (x_{le_j} + C_P c_j) \right]}{c_{avg}} \quad (14)$$

Finally, the span efficiency factor can be calculated by[1]:

$$e = \frac{C_L{}^2}{\pi AR\, C_{D_i}} \quad (15)$$

The method of Lagrange Multipliers can be used to directly solve for the load distribution corresponding to the minimum induced drag coefficient under the constraints of a user-specified C_L and C_M. The constraints are implemented by augmenting the matrix of influence coefficients, which yields the following system of equations:

$$\left[\begin{array}{c|cc} \bar{A} & s_i \cos\theta_i & s_i \cos\theta_i \bar{x}_i \\ & \vdots & \vdots \\ \hline s_j \cos\theta_j \cdots & \multicolumn{2}{c}{\multirow{2}{*}{0}} \\ s_j \cos\theta_j \bar{x}_j \cdots & & \end{array}\right] \left\{\begin{array}{c} \frac{c_n c}{c_{avg}} \\ \hline \lambda_{C_L} \\ \lambda_{C_M} \end{array}\right\} = \left\{\begin{array}{c} 0 \\ \hline 0.5 C_{L_{design}} \\ 0.5 C_{M_{design}} \end{array}\right\} \quad (16)$$

where:

$$\left[\vec{A}\right] = \left[As\right] + \left[As\right]^T \tag{17}$$

$$\bar{x} = x_{cg} - \frac{(x_{le} + C_P c)}{c_{avg}} \tag{18}$$

The system of equations shown above is then solved for the optimum load distribution. When the moment coefficient constraint is not active, the corresponding rows and columns are deleted.

Implementation

The method above was coded in FORTRAN 77. Figure 2 shows a flowchart of the code, entitled *idrag.f*. The code begins by setting up the geometry. This includes calculation of the dihedral angles, local chords, and semi-widths of the lifting elements, and the coordinates of the vortex control points.

At this point, two execution options are available: analysis and design. If the analysis mode is chosen, the code takes the geometry and load distribution as inputs, and calculates the performance parameters as outputs. If the design mode is chosen, the geometry and design conditions are taken as inputs, and the code calculates the load distribution for the minimum induced drag, and returns the performance parameters as outputs. The matrix of influence coefficients and design constraints is formed in subroutine *matrix*, and the resulting matrix is solved via LU decomposition. The LINPACK routine *SGEFS* is used, although any linear solver could be used.[6]

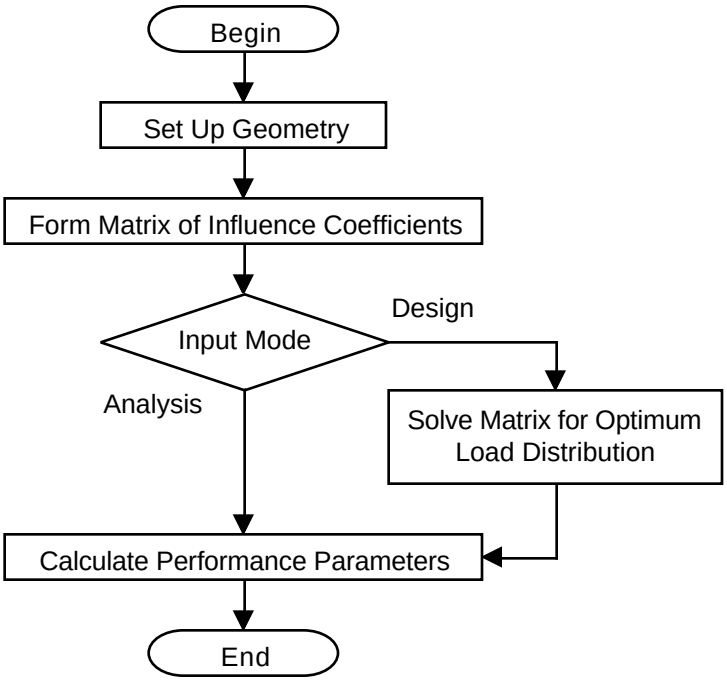

Figure 2: Flowchart of drag code

The user is prompted for the input and output filenames. Note that the comments in the sample input files are ignored, since the unformatted read statements only use the first value on each line of input.

Input

The code has different input formats for the design and analysis modes. The type of input format is specified by the value of the input mode flag on the third line of the input file. A zero designates the design mode, and a one designates the analysis mode.

Design Mode

At this point, the reader should refer to Appendix C, which contains a sample design input file for a wing with a winglet and a conventional horizontal tail. The first line of the input file simply identifies which code the input file belongs to. The second line allows the user to assign a title to the configuration. Note that the title does not have to be the same as the name of the input file. As mentioned above, the input mode flag tells the code whether to expect the input in the design format or the analysis format.

The write flag controls the creation of the output file. A value of 0 specifies that no output file is written, and a value of 1 will cause an output file to be written. If the subroutine is called from within an optimization algorithm, the user should choose not to write an output file in order to alleviate lost time in writing data to the disk for each objective function evaluation.

The symmetry flag is set to zero for asymmetric configurations, and one for symmetric configurations. The code will reflect all of the surfaces about the *xz* plane for symmetric configurations. However, the size of the matrix of influence coefficients is not actually doubled within the code in order to maximize efficiency.

The design lift coefficient is the lift coefficient for the entire configuration. The moment coefficient flag controls the use of the moment coefficient in determining the optimum load distribution. To evaluate the untrimmed load distribution, the moment coefficient flag is set to zero, and it is set to one to evaluate the trimmed load distribution. The design moment coefficient is the moment coefficient about the *cg* for the entire configuration. The location of the center of gravity along the *x* axis is defined by the next input parameter. Since all of the loads are parallel to the *z* axis, the *z* coordinate of the *cg* position is not required.

The *cp* parameter specifies the chordwise location of the center of pressure for all of the airfoil sections. This is used in the moment constraint calculations to determine the moment arms for the distributed loads.

The reference area and reference chord values are input next. This gives the user complete control over the values used to normalize the lift and moment into lift and moment coefficients.

Before defining the aircraft flying surfaces, the number of panels must be specified. The code will then read the inputs for the specified number of panels.

The flying surfaces are then defined by specifying the three-dimensional coordinates of the four corners of a panel. Figure 3 shows a sample input geometry in the *xy* plane. The coordinates of the corners proceed clockwise, starting from the leading edge corner closest to the aircraft centerline. Note that a positive load will always point in a direction given by the right hand rule with a counter-clockwise rotation around the panel. This becomes important in the definition of vertical surfaces such as winglets.

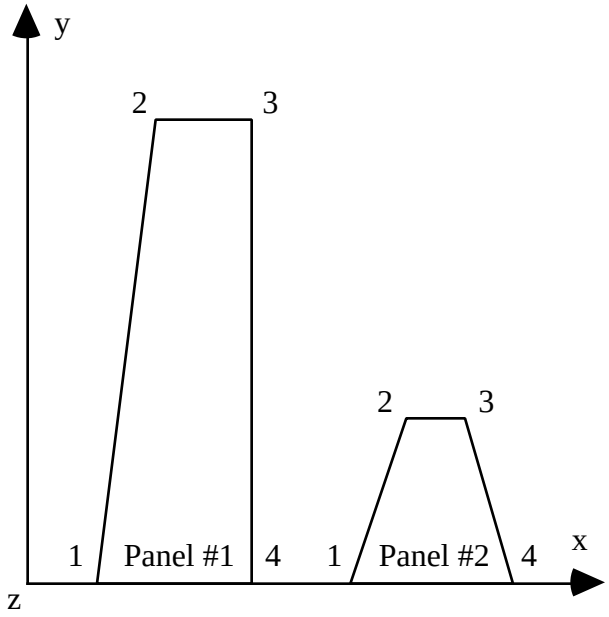

Figure 3: Input geometry

In both input formats, the user has the option of choosing four different types of vortex spacing. This is denoted by assigning one of the following values to the spacing flag parameter:

0	even spacing
1	outboard-packed cosine spacing
2	inboard-packed cosine spacing
3	end-packed cosine spacing

The various spacing options are shown graphically in Figure 4. The use of cosine spacing allows the user to place a higher density of vortices in the area of the flow where the load distribution is changing most rapidly. This allows the code to achieve more accurate results with a smaller number of vortices. This is shown in Figure 5, where the number of vortices on a monoplane wing was varied between 20 and 200 for even spacing, and outboard-packed cosine spacing. In this case the cosine spacing achieved a greater accuracy with just 20 vortices than the even spacing achieved with 200.

The user should exercise some caution when using any of the cosine spacing options. In some cases, cosine spacing can artificially change the local circulation. However, even spacing seems to provide consistent results. Therefore, a sensible approach would be to start with even spacing to establish the basic behavior of the flow, and then if more speed or accuracy is desired, the user can carefully experiment with cosine spacing.

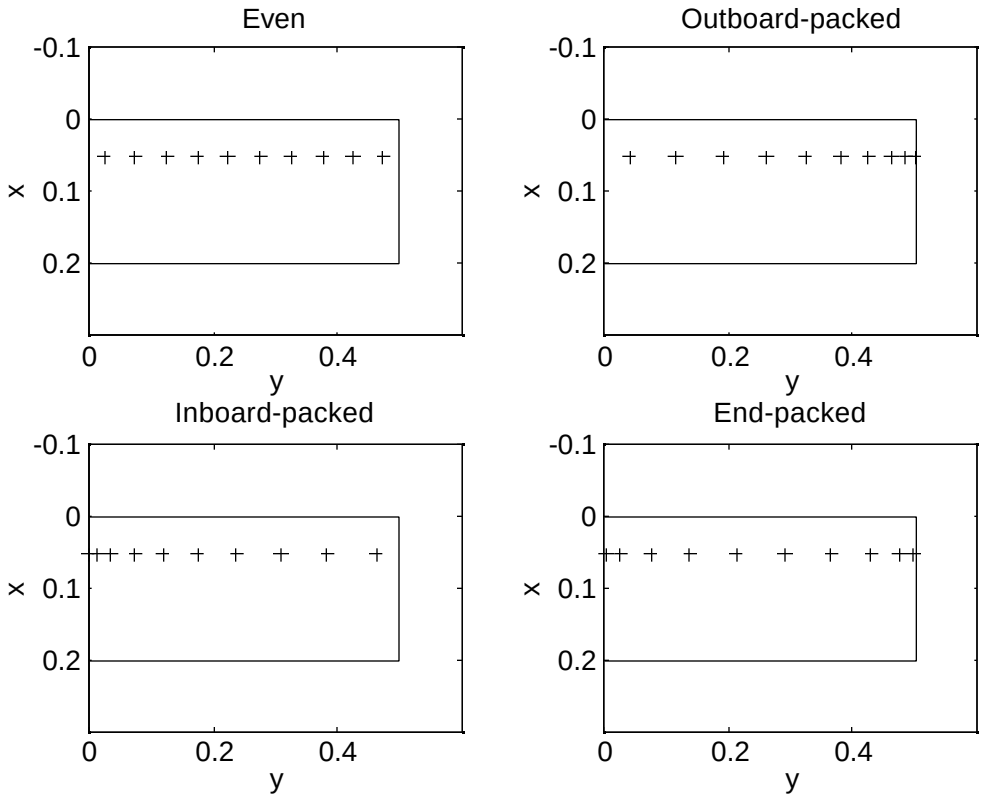

Figure 4: Vortex spacing options

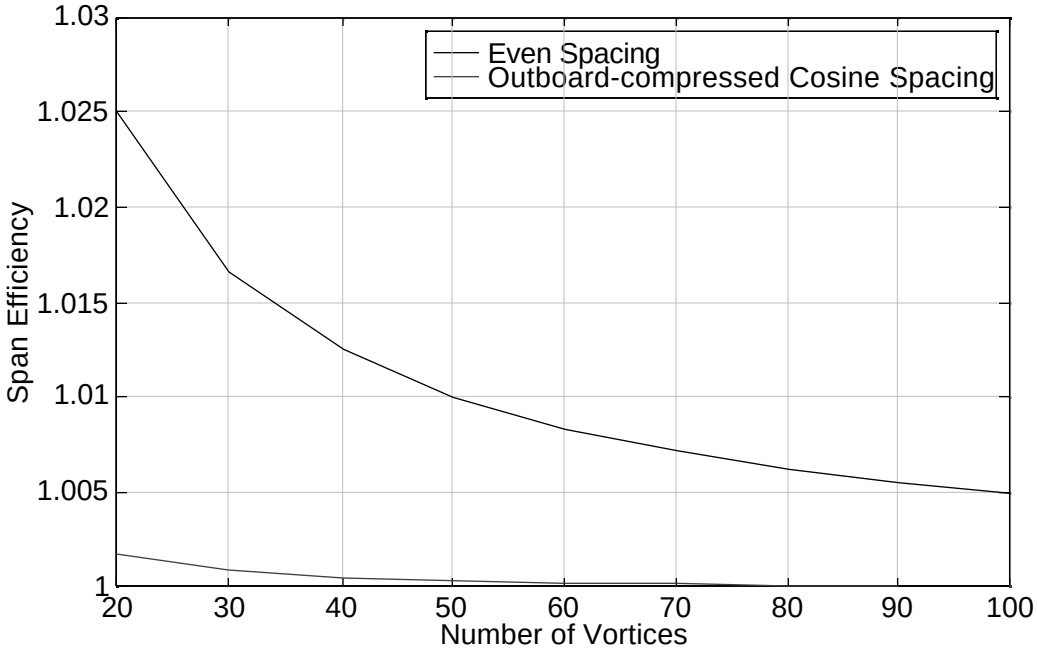

Figure 5: Comparison of spacing options

Analysis Mode

For the analysis mode, the aircraft geometry is input in the same format, but instead of specifying the desired design conditions, the user must specify either the load distribution or the normal force coefficient distribution. The distribution can be specified by two or more points along the length of the panel. The code will linearly interpolate between the points to determine the distribution at a specified number of vortices along the panel.

Appendices E-H show the input and output files for two cases run with the analysis mode. The first case (*asample1*) is for a linear load distribution, with a load of 1 at the root and 0 at the tip. Note that the calculated span efficiency (0.730) agrees with the theoretical span efficiency of 0.73. This is good agreement, especially when considering that only 10 vortices were used. The second case (*asample2*) is for an elliptic load distribution given by 11 points along the span. The calculated span efficiency for this case (1.01) shows good agreement with the theoretical value of 1.00.

Warning about Coplanar Surfaces

The user should exercise some caution when two surfaces are close to being coplanar (i.e. the projections of the surfaces in the *yz* plane are overlapping). In this situation, the matrix of influence coefficients becomes ill-conditioned, and the resulting load distribution becomes noisy and inaccurate. This can be accounted for by adding some space between the two elements in the *yz* plane.

Output

Text

Appendix C shows a sample output file. The first section of the output file is simply a repetition of the input for confirmation. This is followed by a listing of the three-dimensional coordinates of the vortex control points (defined in Figure 1), and the respective load and normal force coefficient distributions at those points. The final section lists the actual lift and moment coefficients, the induced drag coefficient, and the span efficiency factor.

Matlab Plots

Three utility codes have been created which can be used within Matlab: *readidrag.m*, *geom.m*, and *loads.m*. *readidrag.m* reads the parameters from the text output file into the Matlab workspace. It is called by *geom.m* and *loads.m*.

geom.m creates plots of the aircraft configuration. The user can display the perspective view only, or the perspective view along with the top, rear, and side views. The user also has the option of displaying the vortex control points. A set of configuration plots for the wing-with-winglet example is shown in Figure 6.

loads.m creates plots of the load distribution or the normal force coefficient distribution vs. the distance along the panel (note that this is not necessarily *y* or *z*, but rather sqrt($y^2 + z^2$)) for each panel in the aircraft configuration. The code is currently limited to 4 panels, but it can easily be modified to handle more. The load distribution for the wing-with-winglet example is shown in Figure 7.

Help is available within Matlab for all of the codes. To get help on *geom.m*, for example, just type 'help geom' at the Matlab prompt. Note that the codes must either be located in the local directory, or in a directory called ‘matlab’ in the user’s root directory. Matlab automatically places this directory in its path when it starts up.

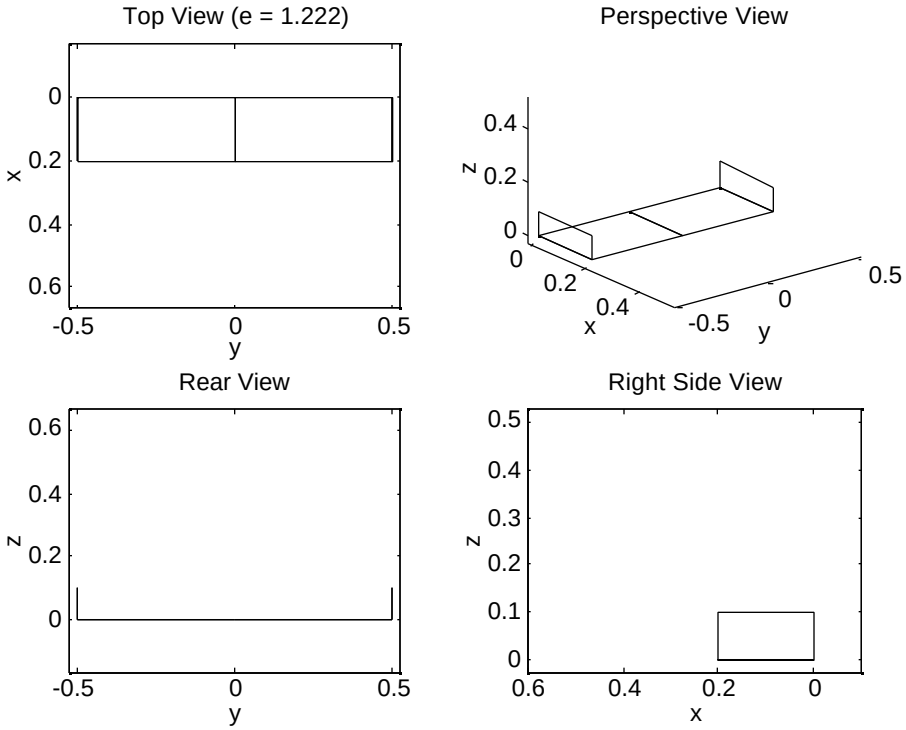

Figure 6: Wing-with-winglet geometry

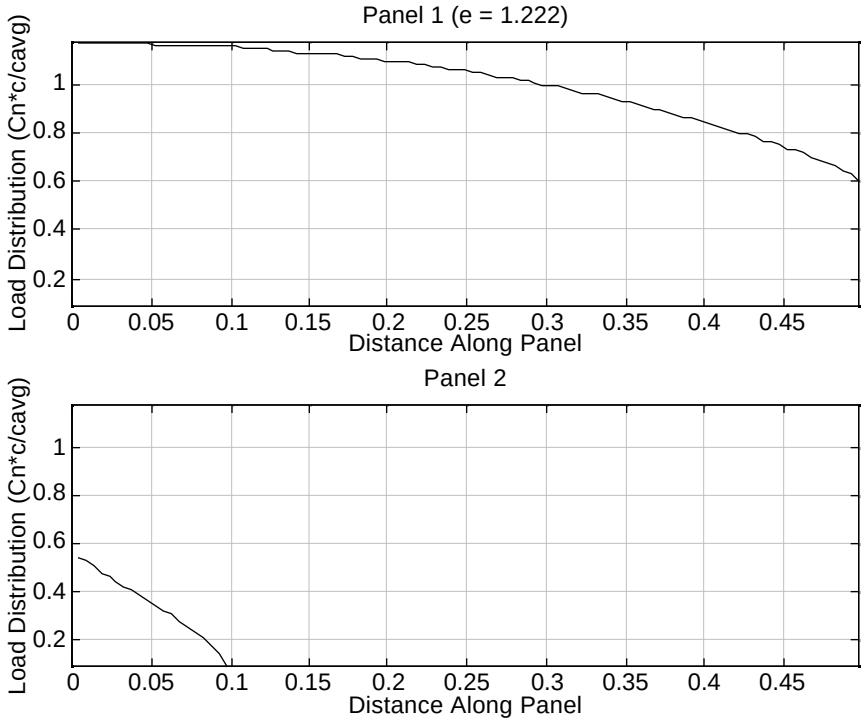

Figure 7: Load distribution for wing-with-winglet

Validation

The induced drag code has been validated with a total of fifteen test cases. The first two test cases correspond to Figures 10 and 11 in Blackwell's paper (a monoplane, and a wing with winglets). The next three cases correspond to Figures 2-4 in Kuhlman's paper. Ten additional test cases were taken from the McMasters and Kroo paper. The configurations for all of the test cases are shown in Figure 8.

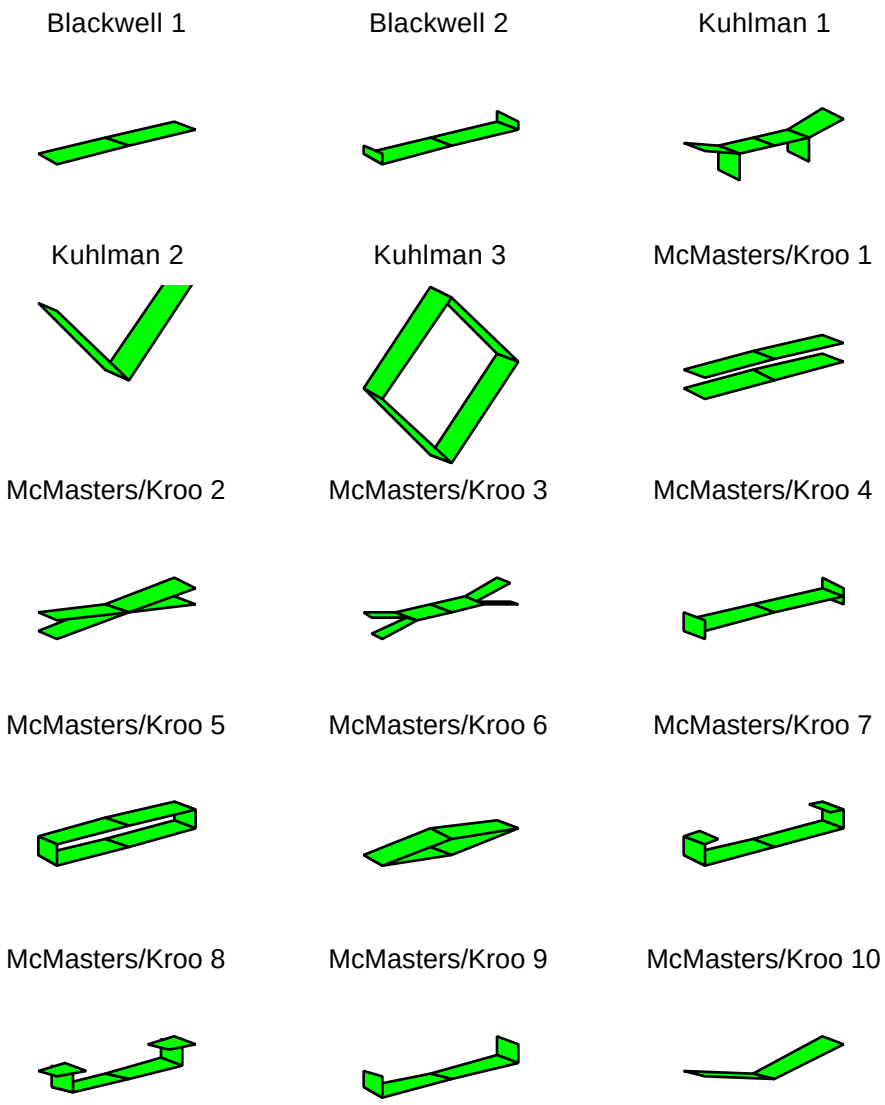

Figure 8: Test case configurations

Figure 9 shows the results obtained for Blackwell's test cases. Although Blackwell's data is not available in numerical form, a comparison of the respective plots shows very good agreement.

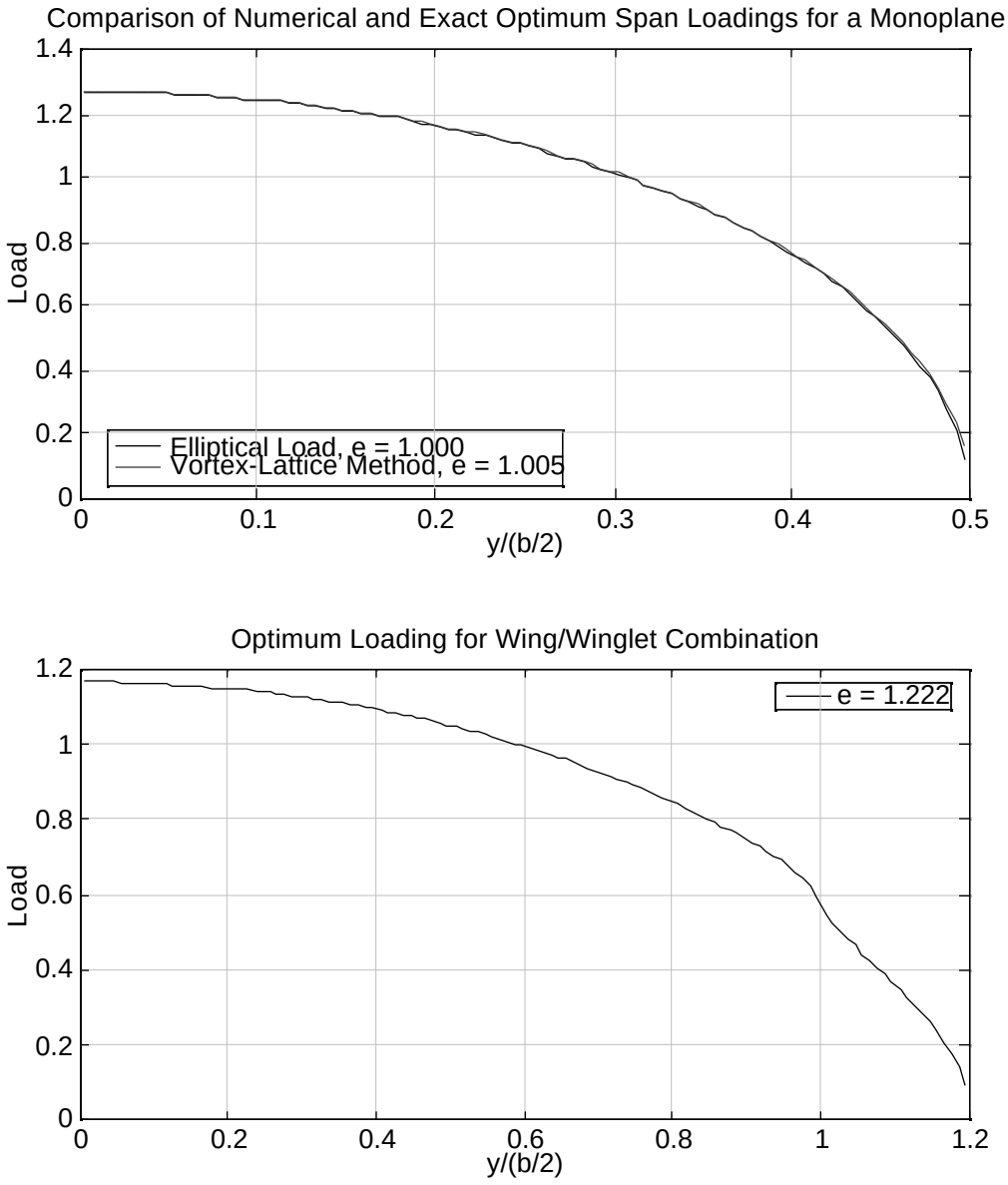

Figure 9: Blackwell test case results

The results for the Kuhlman test cases are shown in Figure 10. A comparison with the original plots shows good agreement.

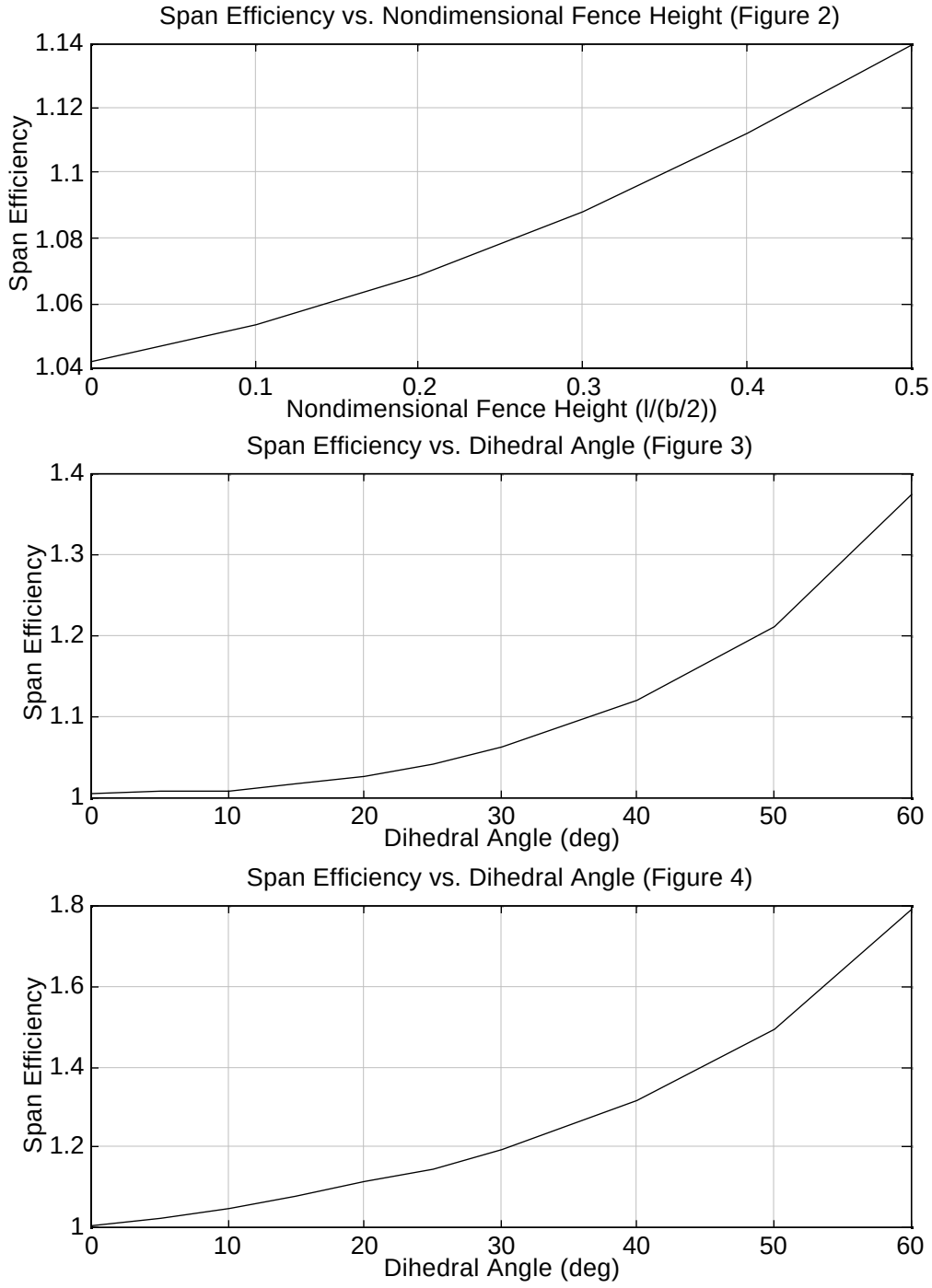

Figure 10: Kuhlman test case results

The McMasters and Kroo paper does not provide exact dimensions for each geometry, but the best guess at the dimensions was made by measuring the drawings with a ruler. The only dimension given is the height-to-span ratio of 0.2. Table 1 presents a comparison of the span efficiency factors as calculated by McMasters/Kroo and the *idrag* code.

Table 1: Comparison of span efficiency factors

Test Case #	McMasters/Kroo	Grasmeyer	% Difference
1	1.36	1.358	0.150
2	1.33	1.331	0.075
3	1.32	1.313	0.530
4	1.38	1.432	3.600
5	1.46	1.484	1.600
6	1.05	1.057	0.660
7	1.45	1.501	5.100
8	1.20	1.224	2.000
9	1.41	1.453	3.000
10	1.03	1.032	0.190

The differences between the results shown in Table 1 are probably due to two factors. First, since the exact planform geometry was not given in the McMasters and Kroo paper, it is not guaranteed that the planforms used in both methods are identical. Second, the number of vortices used for each panel was not given in the McMasters and Kroo paper. For the cases that were run with *idrag*, 200 vortices were used for each panel. A brief study showed that this was sufficient to allow the second decimal place (1/100) to remain unchanged with a small change in the number of vortices. It could be that a small number of vortices were used in the McMasters/Kroo study, resulting in some numerical error.

Accuracy and CPU Time

The results shown above provide confidence that the code is working properly. However, the accuracy of the code is still dependent on the number of vortices used per panel. When exercising the code, the user must strike a compromise between accuracy and run time. In order to accommodate this decision, a brief study was made of the dependence of accuracy and run time on the number of vortices used per panel. The results from this study are shown in Figure 11.

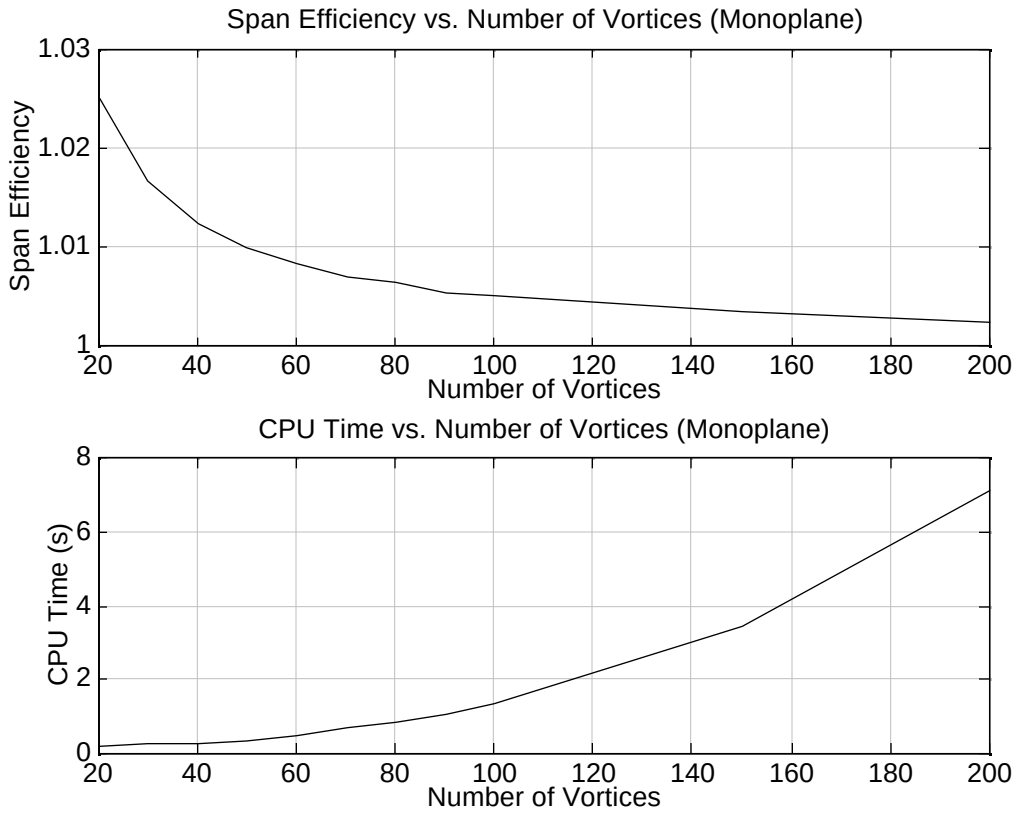

Figure 11: Accuracy and CPU time vs. number of vortices

References

[1]Blackwell, J., "Numerical Method to Calculate the Induced Drag or Optimal Span Loading for Arbitrary Non-Planar Aircraft," NASA SP-405, May 1976.

[2]Lamar, J., "A Vortex Lattice Method for the Mean Camber Shapes of Trimmed Non-Coplanar Planforms with Minimum Vortex Drag," NASA TN-D-8090, June 1976.

[3]Kuhlman, J., and Ku, T., "Numerical Optimization Techniques for Bound Circulation Distribution for Minimum Induced Drag of Nonplanar Wings: Computer Program Documentation," NASA CR-3458, 1982.

[4]Kroo, I., "A General Approach to Multiple Lifting Surface Design and Analysis," AIAA Paper 84-2507, Oct. 1984.

[5]McMasters, J., and Kroo, I., "Advanced Configurations for Very Large Subsonic Transport Airplanes (Some Opportunities for Interdisciplinary Synergism)," NASA Workshop on Potential Impacts of Advanced Aerodynamic Technology on Air Transportation System Productivity, Hampton, VA, June 29-July 31, 1993.

[6]*Math77: Mathematical Subprograms for FORTRAN 77*, Language Systems Corporation, 100 Carpenter Drive, Sterling, VA 20164.

Appendices

Appendix A: idrag Code

```
c///////////////////////////////////////////////////////////////////////
c
c  subroutine idrag
c
c    This subroutine calculates the distribution of the normal force
c  coefficient, induced drag, and span efficiency factor for nonplanar
c  wings composed of multiple panels.  It is based on the paper
c  entitled "Numerical Method to Calculate the Induced Drag or Optimum
c  Loading for Arbitrary Non-planar Aircraft" by James Blackwell, Jr.,
c  from NASA SP-405, Vortex-Lattice Utilization
c
c  Inputs
c
c  outfile      output filename
c  title        title of aircraft configuration
c  input_mode   input mode (0 = design unknown loads, 1 = analyze given loads)
c  write_flag   write flag (0 = no output file, 1 = output file written)
c  sym_flag     symmetry flag (0 = asymmetric, 1 = symmetric)
c  cl_design    design lift coefficient
c  cm_flag      flag for Cm constraint (0 = no constraint, 1 = constraint)
c  cm_design    design pitching moment coefficient (about the cg)
c  xcg          x location of the center of gravity
c  cp           percent chord location of center of pressure for all sections
c  sref         reference area (projected onto xy plane)
c  cavg         average chord of reference surface(s)
c  npanels      number of panels in geometry
c  xc, yc, zc   coordinates of corner points (aircraft reference frame)
c  nvortices    number of vortices per panel
c  spacing_flag vortex spacing flag (0 = equal, 1 = outboard-compressed,
c                2 = inboard-compressed, 3 = end-compressed)
c  load_flag    load flag (0 = cn input, 1 = load (cn*c/cavg) input)
c  loads        vector of loads for analysis mode
c
c  Outputs
c
c  cd_induced   induced drag coefficient
c
c  Internal Variables
c
c  abar         augmented influence coefficient matrix (Lamar)
c  ar           aspect ratio of reference surface
c  b            vector of constraints; overwritten with loads vector
c  bpanel       span of panel (projected onto xy plane)
c  bref         reference span (projected onto xy plane)
c  c            local chord of lifting element
c  cl_actual    actual lift coefficient (obtained via integration)
c  cm_actual    actual pitching moment coefficient (about the cg)
c  cn           vector of normal force coefficients
c  d            dummy output from LU decomposition subroutine
c  e            span efficiency factor
c  i            index
c  info         SGEFS information flag
c  ipvt         SGEFS output vector of row interchanges
c  j            index
```

```
c  lda          leading dimension of abar matrix
c  ldb          leading dimension of b matrix
c  n            actual size of abar matrix
c  nb           number of columns in b matrix
c  nconstraints number of constraints (parameter)
c  npanels_max  maximum number of panels in geometry
c  nvortices_max  maximum number of vortices per panel
c  nvortices_tot  total number of vortices
c  s            vortex semi-width (non-dimensional)
c  semi_width   temporary parameter for calculating vortex semi-width
c  sp           vortex semi-width (dimensional)
c  spacing      vortex spacing
c  theta        vortex dihedral angle
c  x, y, z      coordinates of vortices (aircraft reference frame)
c  xle, xte     coordinates of LE and TE corresponding to each vortex
c
c  Created by:  Joel Grasmeyer
c  Last Modified:  02/04/97
c  Version:  1.0
c
c//////////////////////////////////////////////////////////////////////

      subroutine idrag(outfile,title,input_mode,write_flag,sym_flag,
     & cl_design,cm_flag,cm_design,xcg,cp,sref,cavg,npanels,xc,yc,zc,
     & nvortices,spacing_flag,load_flag,loads,cd_induced)

      implicit none
      integer npanels_max, nvortices_max, nconstraints
      parameter(npanels_max=5, nvortices_max=200, nconstraints=2)

      character*72 outfile, title
      integer nvortices_tot, n, i, j, cm_flag, input_mode,
     & write_flag, sym_flag, npanels, nvortices(npanels_max),
     & spacing_flag(npanels_max), load_flag, lda, ldb, nb, info,
     & ipvt(npanels_max*nvortices_max+nconstraints)
      real cl_design, sref, bref, cavg, x(npanels_max*nvortices_max),
     & y(npanels_max*nvortices_max), z(npanels_max*nvortices_max),
     & c(npanels_max*nvortices_max), s(npanels_max*nvortices_max),
     & sp(npanels_max*nvortices_max), theta(npanels_max*nvortices_max),
     & a(npanels_max*nvortices_max,npanels_max*nvortices_max), d, pi,
     & indx(npanels_max*nvortices_max+nconstraints), cm_design, xcg,
     & b(npanels_max*nvortices_max+nconstraints), cp, cm_actual,
     & abar(npanels_max*nvortices_max+nconstraints,
     & npanels_max*nvortices_max+nconstraints), cl_actual, e,
     & xle(npanels_max*nvortices_max), cd_induced, ar,
     & cn(npanels_max*nvortices_max), xc(npanels_max,4),
     & loads(npanels_max*nvortices_max), yc(npanels_max,4), bpanel,
     & spacing, zc(npanels_max,4), semi_width,
     & xte(npanels_max*nvortices_max)

      pi = acos(-1.)

c Write input data to output file for confirmation
      if (write_flag .eq. 1) then
        open(11,file=outfile)
        write(11,"('idrag output file')")
        write(11,"(a72)") title
        write(11,101) input_mode, '= input mode'
        write(11,101) write_flag, '= write flag'
        write(11,101) sym_flag, '= symmetry flag'
        write(11,102) cl_design, '= design lift coefficient'
```

```
        write(11,101) cm_flag, '= moment coefficient flag'
        write(11,102) cm_design, '= design moment coefficient'
        write(11,102) xcg, '= x cg position'
        write(11,102) cp, '= center of pressure for airfoil sections'
        write(11,102) sref, '= reference area'
        write(11,102) cavg, '= reference chord'
        write(11,101) npanels, '= number of panels'
        do i=1,npanels
          write(11,*)
          write(11,103) '    x        y        z    for panel ', i
          do j=1,4
            write(11,104) xc(i,j), yc(i,j), zc(i,j)
          end do
          write(11,101) nvortices(i), '= number of vortices'
          write(11,101) spacing_flag(i), '= vortex spacing flag'
        end do
  101   format(2x, i3, 4x, a)
  102   format(f8.2, 1x, a)
  103   format(a, i1)
  104   format(3(f8.2, 1x))
      end if

c Calculate reference span
      bref = sref/cavg

c Begin vortex loop
      nvortices_tot = 0
      do i=1,npanels
        do j=1,nvortices(i)
          nvortices_tot = nvortices_tot + 1

c Calculate dihedral angles of lifting elements
          if (yc(i,2) .eq. yc(i,1)) then
            theta(nvortices_tot) = pi/2.
          else
            theta(nvortices_tot) = atan((zc(i,2) - zc(i,1))/
     &       (yc(i,2) - yc(i,1)))
          end if

c Calculate spacing (0 = equal, 1 = outboard-compressed, 2 = inboard-
c   compressed, 3 = end-compressed) and panel semi-widths
          if (spacing_flag(i) .eq. 0) then
            spacing = (j - 0.5)/nvortices(i)
            semi_width = 0.5/nvortices(i)
          elseif (spacing_flag(i) .eq. 1) then
            spacing = sin(pi/2.*(j - 0.5)/nvortices(i))
            semi_width = 0.5*(sin(pi/2.*j/nvortices(i)) -
     &       sin(pi/2.*(j - 1.)/nvortices(i)))
          elseif (spacing_flag(i) .eq. 2) then
            spacing = 1 - cos(pi/2.*(j - 0.5)/nvortices(i))
            semi_width = 0.5*(-cos(pi/2.*j/nvortices(i)) +
     &       cos(pi/2.*(j - 1.)/nvortices(i)))
          elseif (spacing_flag(i) .eq. 3) then
            spacing = (1 - cos(pi*(j - 0.5)/nvortices(i)))/2.
            semi_width = 0.25*(-cos(pi*j/nvortices(i)) +
     &       cos(pi*(j - 1.)/nvortices(i)))
          end if

          sp(nvortices_tot) = semi_width*sqrt((yc(i,2) -
     &     yc(i,1))**2 + (zc(i,2) - zc(i,1))**2)
          s(nvortices_tot) = 2.*sp(nvortices_tot)/bref
```

```
c Calculate coordinates of vortices and local chords
          y(nvortices_tot) = yc(i,1) + spacing*(yc(i,2) - yc(i,1))
          z(nvortices_tot) = zc(i,1) + spacing*(zc(i,2) - zc(i,1))
          xle(nvortices_tot) = xc(i,1) + spacing*(xc(i,2) - xc(i,1))
          xte(nvortices_tot) = xc(i,4) + spacing*(xc(i,3) - xc(i,4))
          c(nvortices_tot) = (xte(nvortices_tot) -
     &     xle(nvortices_tot))
          x(nvortices_tot) = xle(nvortices_tot) + 0.25*c(nvortices_tot)
        end do
      end do

c Form matrix of influence coefficients with constraints
      call matrix(sym_flag,cl_design,cm_flag,cm_design,xcg,cp,
     & nvortices_tot,y,z,theta,s,sp,c,cavg,xle,a,abar,b)

c Design mode:  calculate loading for minimum induced drag
      if (input_mode .eq. 0) then
        if (cm_flag .eq. 0) then
          n = nvortices_tot+1
        else
          n = nvortices_tot+2
        end if
        lda = npanels_max*nvortices_max+nconstraints
        ldb = lda
        nb = 1

c Calculate the loads (note that the b vector is an input and output)
        call sgefs(abar,lda,n,b,ldb,nb,ipvt,info)

c Analysis mode:  calculate performance for given loading
      else

c If Cn values were input, convert to loads (Cn*c/cavg)
        if (load_flag .eq. 0) then
          do i=1,nvortices_tot
            loads(i) = loads(i)*c(i)/cavg
          end do
        end if

c Set b vector to given loads
        do i=1,nvortices_tot
          b(i) = loads(i)
        end do
      end if

c Calculate actual lift coefficient, induced drag, and span efficiency
      cl_actual = 0.
      cm_actual = 0.
      cd_induced = 0.
      do i=1,nvortices_tot
        if (sym_flag .eq. 1) then
          cl_actual = cl_actual + 2.*b(i)*s(i)*cos(theta(i))
          cm_actual = cm_actual + 2.*b(i)*s(i)*cos(theta(i))*
     &     (xcg - (xle(i) + cp*c(i)))/cavg
        else
          cl_actual = cl_actual + b(i)*s(i)*cos(theta(i))
          cm_actual = cm_actual + b(i)*s(i)*cos(theta(i))*
     &     (xcg - (xle(i) + cp*c(i)))/cavg
        end if
        cn(i) = b(i)*cavg/c(i)
```

```
        do j=1,nvortices_tot
          if (sym_flag .eq. 1) then
            cd_induced = cd_induced + b(i)*b(j)*s(i)*a(i,j)
          else
            cd_induced = cd_induced + 0.5*b(i)*b(j)*s(i)*a(i,j)
          end if
        end do
      end do
      ar = bref**2/sref
      e = cl_actual**2/(pi*ar*cd_induced)

c Write vortex positions, load distribution and performance to output file
      if (write_flag .eq. 1) then
        write(11,*)
        write(11,110) '    i   x        y        z        load     cn'
        do i=1,nvortices_tot
          write(11,111) i, x(i), y(i), z(i), b(i), cn(i)
        end do
        write(11,*)
        write(11,112) cl_actual, '= actual lift coefficient'
        write(11,112) cm_actual, '= actual moment coefficient'
        write(11,112) cd_induced, '= induced drag coefficient'
        write(11,112) e, '= span efficiency factor'
  110   format(a)
  111   format(x, i4, 5(1x, f8.4))
  112   format(4x, f7.5, 1x, a)
        close(11)
      end if

      return
      end


c//////////////////////////////////////////////////////////////////////
c
c  subroutine matrix
c
c    This subroutine forms the matrix of influence coefficients from
c  the geometry information provided by the idrag subroutine.  The
c  matrix equation (Ax = B) is then solved via the subroutines ludcmp
c  and lubksb.
c
c  Inputs
c
c  sym_flag     symmetry flag (0 = asymmetric, 1 = symmetric)
c  cl_design    design lift coefficient
c  cm_flag      flag for Cm constraint (0 = no constraint, 1 = constraint)
c  cm_design    design pitching moment coefficent (about the cg)
c  xcg          x location of the center of gravity
c  cp           percent chord location of center of pressure for all sections
c  nvortices_tot  total number of vortices
c  y, z         coordinates of vortices (aircraft reference frame)
c  theta        vortex dihedral angle
c  s            vortex semi-width (non-dimensional)
c  sp           vortex semi-width (dimensional)
c  c            local chord of lifting element
c  cavg         average chord of reference surface(s)
c  xle          coordinates of LE corresponding to each vortex
c
c  Outputs
c
```

```
c  a            influence coefficient matrix (Blackwell)
c  abar         augmented influence coefficient matrix (Lamar)
c  b            vector of constraints
c
c  Internal Variables
c
c  a1           matrix of influence coefficients for same-side vortices
c  a2           matrix of influence coefficients for image vortices
c  i            index
c  j            index
c  nconstraints number of constraints (parameter)
c  npanels_max  maximum number of panels in geometry
c  nvortices_max  maximum number of vortices per panel
c  r1, r2       vortex influence radii squared
c  yp, zp       coordinates of vortices (vortex reference frame)
c
c  Created by:  Joel Grasmeyer
c  Last Modified:  12/11/96
c
c*********************************************************************

      subroutine matrix(sym_flag,cl_design,cm_flag,cm_design,xcg,cp,
     & nvortices_tot,y,z,theta,s,sp,c,cavg,xle,a,abar,b)

      implicit none
      integer npanels_max, nvortices_max, nconstraints
      parameter(npanels_max=5,nvortices_max=200,nconstraints=2)

      integer i, j, nvortices_tot, cm_flag, sym_flag
      real pi, yp, zp, y(npanels_max*nvortices_max), r1, r2, a1,
     & z(npanels_max*nvortices_max), theta(npanels_max*nvortices_max),
     & s(npanels_max*nvortices_max), sp(npanels_max*nvortices_max),
     & a2, a(npanels_max*nvortices_max,npanels_max*nvortices_max),
     & abar(npanels_max*nvortices_max+nconstraints,
     & npanels_max*nvortices_max+nconstraints), cp, cm_design, xcg,
     & b(npanels_max*nvortices_max+nconstraints), cavg, cl_design,
     & c(npanels_max*nvortices_max), xle(npanels_max*nvortices_max)

      pi = acos(-1.)

c Form the matrix of influence coefficients
      do i=1,nvortices_tot
        do j=1,nvortices_tot

c First, calculate the effects of the same-side vortices
          yp = (y(i) - y(j))*cos(theta(j)) +
     &         (z(i) - z(j))*sin(theta(j))
          zp = -(y(i) - y(j))*sin(theta(j)) +
     &          (z(i) - z(j))*cos(theta(j))
          r1 = zp**2 + (yp - sp(j))**2
          r2 = zp**2 + (yp + sp(j))**2

          a1 = ((yp - sp(j))/r1 - (yp + sp(j))/r2)*cos(theta(i)
     &     - theta(j)) + (zp/r1 - zp/r2)*sin(theta(i) - theta(j))

c If the configuration is symmetric, change the sign of y(j) and
c theta(j) to account for image vortices
          if (sym_flag .eq. 1) then
            yp = (y(i) + y(j))*cos(-theta(j)) +
     &           (z(i) - z(j))*sin(-theta(j))
            zp = -(y(i) + y(j))*sin(-theta(j)) +
```

```
     &            (z(i) - z(j))*cos(-theta(j))
            r1 = zp**2 + (yp - sp(j))**2
            r2 = zp**2 + (yp + sp(j))**2
            a2 = ((yp - sp(j))/r1 - (yp + sp(j))/r2)*cos(theta(i)
     &       + theta(j)) + (zp/r1 - zp/r2)*sin(theta(i) + theta(j))
          else
            a2 = 0.
          end if

c Add the two influences to form the total influence coefficients
          a(i,j) = -cavg/(4.*pi)*(a1 + a2)

        end do
      end do

c Implement method of Lagrange multipliers
      do i=1,nvortices_tot
        do j=1,nvortices_tot
          abar(i,j) = a(i,j)*s(i) + a(j,i)*s(j)
        end do
        b(i) = 0.
      end do

c Augment influence coefficient matrix with Cl constraint
      do i=1,nvortices_tot
        abar(i,nvortices_tot+1) = s(i)*cos(theta(i))
      end do
      do j=1,nvortices_tot
        abar(nvortices_tot+1,j) = s(j)*cos(theta(j))
      end do
      abar(nvortices_tot+1,nvortices_tot+1) = 0.
      if (sym_flag .eq. 1) then
        b(nvortices_tot+1) = cl_design/2.
      else
        b(nvortices_tot+1) = cl_design
      end if

c Augment matrix with Cm constraint if cm_flag = 1
      if (cm_flag .eq. 1) then
        do i=1,nvortices_tot
          abar(i,nvortices_tot+2) = s(i)*cos(theta(i))*
     &     (xcg - (xle(i) + cp*c(i)))/cavg
        end do
        do j=1,nvortices_tot
          abar(nvortices_tot+2,j) = s(j)*cos(theta(j))*
     &     (xcg - (xle(j) + cp*c(j)))/cavg
        end do
        abar(nvortices_tot+1,nvortices_tot+2) = 0.
        abar(nvortices_tot+2,nvortices_tot+1) = 0.
        abar(nvortices_tot+2,nvortices_tot+2) = 0.
        if (sym_flag .eq. 1) then
          b(nvortices_tot+2) = cm_design/2.
        else
          b(nvortices_tot+2) = cm_design
        end if
      end if

      return
      end
```

Appendix B: idragin Code

```
c////////////////////////////////////////////////////////////////////
c
c  program idragin
c
c    This program reads the design or analysis input file, and calls
c  the idrag subroutine to calculate the performance of the configuration.
c
c  Program Execution
c
c    The user is prompted for the input and output filenames.  Note that
c  the comments in the sample input files are ignored, since the
c  unformatted read statements only use the first value on each line of
c  input.
c
c  Variable Definitions
c
c  cd_induced   induced drag coefficient
c  cl_design    design lift coefficient
c  cm_flag      flag for Cm constraint (0 = no constraint, 1 = constraint)
c  cm_design    design pitching moment coefficent (about the cg)
c  cp           percent chord location of center of pressure for all sections
c  cavg         average chord (cavg = sref/bref)
c  d            distance along span for load distribution
c  header       header which identifies the input and output files
c  i            index
c  infile       input filename
c  input_mode   input mode (0 = design unknown loads, 1 = analyze given loads)
c  j            index
c  k            index
c  loads        vector of loads for analysis mode
c  load_flag    load flag (0 = cn input, 1 = load (cn*c/cavg) input)
c  load_input   input loads specified at given spanwise stations
c  load_station percent span locations of specified loads in the range [0,1]
c  nloads       number of loads specified per panel for analysis mode input
c  npanels      number of panels in geometry
c  npanels_max  maximum number of panels in geometry
c  nvortices    number of vortices per panel
c  nvortices_max  maximum number of vortices per panel
c  outfile      output filename
c  p            linear interpolation parameter
c  spacing_flag vortex spacing flag (0 = equal, 1 = outboard-compressed,
c                2 = inboard-compressed, 3 = end-compressed)
c  sref         reference area
c  sym_flag     symmetry flag (0 = asymmetric, 1 = symmetric)
c  temp_load    temporary load vector for analysis mode
c  title        title of aircraft configuration
c  write_flag   write flag (0 = no output file, 1 = output file written)
c  xc, yc, zc   coordinates of corner points (aircraft reference frame)
c  xcg          x location of the center of gravity
c
c  Created by:  Joel Grasmeyer
c  Last Modified:  02/04/97
c
c////////////////////////////////////////////////////////////////////

      program idragin
```

```
      implicit none
      integer npanels_max, nvortices_max
      parameter(npanels_max=5,nvortices_max=200)

      character*72 infile, outfile, header, title
      integer npanels, nvortices(npanels_max),
     & i, j, input_mode, cm_flag, write_flag, sym_flag,
     & nloads(npanels_max), load_flag, k, spacing_flag(npanels_max)
      real cl_design, sref, cm_design, xcg, cp, cavg, cd_induced,
     & xc(npanels_max,4), yc(npanels_max,4), zc(npanels_max,4),
     & load_station(npanels_max,nvortices_max), d(nvortices_max),
     & load_input(npanels_max,nvortices_max), p(nvortices_max),
     & temp_load(npanels_max,nvortices_max),
     & loads(npanels_max*nvortices_max)

c Get input and output filenames
      write(6,*) 'Enter input filename within single quotes: '
      read(5,*) infile
      write(6,*) 'Enter output filename within single quotes: '
      read(5,*) outfile

c Read input that is common to both modes
      open(10,file=infile)
      read(10,"(a72)") header
      read(10,"(a72)") title
      read(10,*) input_mode
      read(10,*) write_flag
      read(10,*) sym_flag

c Read input file for design mode
      if (input_mode .eq. 0) then
        read(10,*) cl_design
        read(10,*) cm_flag
        read(10,*) cm_design
        read(10,*) xcg
        read(10,*) cp
        read(10,*) sref
        read(10,*) cavg
        read(10,*) npanels
        do i=1,npanels
          do j=1,4
            read(10,*) xc(i,j), yc(i,j), zc(i,j)
          end do
          read(10,*) nvortices(i)
          read(10,*) spacing_flag(i)
        end do

c Initialize variables that are not used in the design mode
        load_flag = 0
        do i=1,npanels_max*nvortices_max
          loads(i) = 0.
        end do

c Read input file for analysis mode
      else
        read(10,*) load_flag
        read(10,*) xcg
        read(10,*) cp
        read(10,*) sref
        read(10,*) cavg
        read(10,*) npanels
```

```
        do i=1,npanels
          do j=1,4
            read(10,*) xc(i,j), yc(i,j), zc(i,j)
          end do
          read(10,*) nvortices(i)
          read(10,*) spacing_flag(i)
          read(10,*) nloads(i)
          do k=1,nloads(i)
            read(10,*) load_station(i,k), load_input(i,k)
          end do
        end do

c Initialize variables that are not used in the analysis mode
        cl_design = 0.
        cm_flag = 0
        cm_design = 0.

c Perform linear interpolation on load input
        do i=1,npanels
          do j=1,nvortices(i)
            d(j) = (j - 0.5)/nvortices(i)
          end do
          j = 1
          k = 2
          do while (j .le. nvortices(i))
            do while (d(j) .le. load_station(i,k) .and.
     &       j .le. nvortices(i))
              p(j) = (d(j) - load_station(i,k-1))/
     &         (load_station(i,k) - load_station(i,k-1))
              temp_load(i,j) = load_input(i,k-1) + p(j)*
     &         (load_input(i,k) - load_input(i,k-1))
              j = j + 1
            end do
            k = k + 1
          end do
        end do

c Vectorize loads
        k = 0
        do i=1,npanels
          do j=1,nvortices(i)
            k = k + 1
            loads(k) = temp_load(i,j)
          end do
        end do

      end if

c Close input file
      close(10)

c Call drag subroutine to calculate performance
       call idrag(outfile,title,input_mode,write_flag,sym_flag,
     & cl_design,cm_flag,cm_design,xcg,cp,sref,cavg,npanels,xc,yc,zc,
     & nvortices,spacing_flag,load_flag,loads,cd_induced)

       end
```

Appendix C: Sample Design Input File (dsample.in)

```
idrag input file
winglet
0               input mode
1               write flag
1               symmetry flag
1.0             cl_design
1               cm_flag
0               cm_design
0.03            x cg position
0.25            center of pressure for airfoil sections
0.2             reference area
0.2             reference chord
3               number of panels
0   0   0       x,y,z for 4 corners of panel 1
0   0.5 0
0.2 0.5 0
0.2 0   0
10              number of vortices for panel 1
0               vortex spacing for panel 1
0   0.5 0       x,y,z for 4 corners of panel 2
0   0.5 0.1
0.2 0.5 0.1
0.2 0.5 0
5               number of vortices for panel 2
0               vortex spacing for panel 2
1   0   0.1     x,y,z for 4 corners of panel 3
1   0.2 0.1
1.1 0.2 0.1
1.1 0   0.1
6               number of vortices for panel 3
0               vortex spacing for panel 3
```

Appendix D: Sample Design Output File (dsample.idrag)

```
idrag output file
winglet
    0    = input mode
    1    = write flag
    1    = symmetry flag
    1.00 = design lift coefficient
    1    = moment coefficient flag
    0.00 = design moment coefficient
    0.03 = x cg position
    0.25 = center of pressure for airfoil sections
    0.20 = reference area
    0.20 = reference chord
    3    = number of panels

    x        y        z    for panel 1
    0.00     0.00     0.00
    0.00     0.50     0.00
    0.20     0.50     0.00
    0.20     0.00     0.00
   10    = number of vortices
    0    = vortex spacing flag

    x        y        z    for panel 2
    0.00     0.50     0.00
    0.00     0.50     0.10
    0.20     0.50     0.10
    0.20     0.50     0.00
    5    = number of vortices
    0    = vortex spacing flag

    x        y        z    for panel 3
    1.00     0.00     0.10
    1.00     0.20     0.10
    1.10     0.20     0.10
    1.10     0.00     0.10
    6    = number of vortices
    0    = vortex spacing flag

    i   x        y        z        load     cn
    1   0.0500   0.0250   0.0000   1.1867   1.1867
    2   0.0500   0.0750   0.0000   1.1756   1.1756
    3   0.0500   0.1250   0.0000   1.1534   1.1534
    4   0.0500   0.1750   0.0000   1.1205   1.1205
    5   0.0500   0.2250   0.0000   1.0784   1.0784
    6   0.0500   0.2750   0.0000   1.0287   1.0287
    7   0.0500   0.3250   0.0000   0.9709   0.9709
    8   0.0500   0.3750   0.0000   0.9041   0.9041
    9   0.0500   0.4250   0.0000   0.8292   0.8292
   10   0.0500   0.4750   0.0000   0.7574   0.7574
   11   0.0500   0.5000   0.0100   0.4581   0.4581
   12   0.0500   0.5000   0.0300   0.4496   0.4496
   13   0.0500   0.5000   0.0500   0.3795   0.3795
   14   0.0500   0.5000   0.0700   0.2967   0.2967
   15   0.0500   0.5000   0.0900   0.1938   0.1938
   16   1.0250   0.0167   0.1000  -0.0642  -0.1284
   17   1.0250   0.0500   0.1000  -0.0622  -0.1244
   18   1.0250   0.0833   0.1000  -0.0581  -0.1162
```

```
19   1.0250   0.1167   0.1000  -0.0517  -0.1034
20   1.0250   0.1500   0.1000  -0.0425  -0.0851
21   1.0250   0.1833   0.1000  -0.0290  -0.0579

 1.00000 = actual lift coefficient
 0.00000 = actual moment coefficient
 0.05008 = induced drag coefficient
 1.27132 = span efficiency factor
```

Appendix E: Sample Analysis Input File (asample1.in)

```
idrag input file
asample1
1                input mode
1                write flag
1                symmetry flag
1                load flag
0.               x cg position
0.25             center of pressure for airfoil sections
0.15             reference area
0.15             reference chord
1                number of panels
0   0   0        x,y,z for 4 corners of panel 1
0   0.5 0
0.2 0.5 0
0.2 0   0
10               number of vortices for panel 1
0                vortex spacing for panel 1
2                number of loads for panel 1
0 1              load station 1, load 1, for panel 1
1 0              load station 2, load 2, for panel 1
```

The load station is a percent span location of a specified load. It can take values in the range [0,1]. Any number of load station/load pairs can be specified, as long as two are located at 0 and 1.

Appendix F: Sample Analysis Output File (asample1.idrag)

```
idrag output file
asample1
    1    = input mode
    1    = write flag
    1    = symmetry flag
    0.00 = design lift coefficient
    0    = moment coefficient flag
    0.00 = design moment coefficient
    0.00 = x cg position
    0.25 = center of pressure for airfoil sections
    0.15 = reference area
    0.15 = reference chord
    1    = number of panels

    x        y        z    for panel 1
    0.00     0.00     0.00
    0.00     0.50     0.00
    0.20     0.50     0.00
    0.20     0.00     0.00
   10    = number of vortices
    0    = vortex spacing flag

    i   x        y        z        load     cn
    1   0.0500   0.0250   0.0000   0.9500   0.7125
    2   0.0500   0.0750   0.0000   0.8500   0.6375
    3   0.0500   0.1250   0.0000   0.7500   0.5625
    4   0.0500   0.1750   0.0000   0.6500   0.4875
    5   0.0500   0.2250   0.0000   0.5500   0.4125
    6   0.0500   0.2750   0.0000   0.4500   0.3375
    7   0.0500   0.3250   0.0000   0.3500   0.2625
    8   0.0500   0.3750   0.0000   0.2500   0.1875
    9   0.0500   0.4250   0.0000   0.1500   0.1125
   10   0.0500   0.4750   0.0000   0.0500   0.0375

    0.50000 = actual lift coefficient
    -.16667 = actual moment coefficient
    0.01636 = induced drag coefficient
    0.72964 = span efficiency factor
```

Appendix G: Sample Analysis Input File (asample2.in)

```
idrag input file
asample2
1                 input mode
1                 write flag
1                 symmetry flag
1                 load flag
0.                x cg position
0.25              center of pressure for airfoil sections
0.15              reference area
0.15              reference chord
1                 number of panels
0   0   0         x,y,z for 4 corners of panel 1
0   0.5 0
0.2 0.5 0
0.2 0   0
10                number of vortices for panel 1
0                 vortex spacing for panel 1
11                number of loads for panel 1
0.0000    1.0000  load station  1, load  1, for panel 1
0.1000    0.9950  load station  2, load  2, for panel 1
0.2000    0.9798  load station  3, load  3, for panel 1
0.3000    0.9539  load station  4, load  4, for panel 1
0.4000    0.9165  load station  5, load  5, for panel 1
0.5000    0.8660  load station  6, load  6, for panel 1
0.6000    0.8000  load station  7, load  7, for panel 1
0.7000    0.7141  load station  8, load  8, for panel 1
0.8000    0.6000  load station  9, load  9, for panel 1
0.9000    0.4359  load station 10, load 10, for panel 1
1.0000    0.0000  load station 11, load 11, for panel 1
```

Appendix H: Sample Analysis Output File (asample2.idrag)

```
idrag output file
asample2
    1    = input mode
    1    = write flag
    1    = symmetry flag
    0.00 = design lift coefficient
    0    = moment coefficient flag
    0.00 = design moment coefficient
    0.00 = x cg position
    0.25 = center of pressure for airfoil sections
    0.15 = reference area
    0.15 = reference chord
    1    = number of panels

    x        y        z    for panel 1
    0.00     0.00     0.00
    0.00     0.50     0.00
    0.20     0.50     0.00
    0.20     0.00     0.00
   10    = number of vortices
    0    = vortex spacing flag

    i   x        y        z        load     cn
    1   0.0500   0.0250   0.0000   0.9975   0.7481
    2   0.0500   0.0750   0.0000   0.9874   0.7405
    3   0.0500   0.1250   0.0000   0.9668   0.7251
    4   0.0500   0.1750   0.0000   0.9352   0.7014
    5   0.0500   0.2250   0.0000   0.8913   0.6684
    6   0.0500   0.2750   0.0000   0.8330   0.6248
    7   0.0500   0.3250   0.0000   0.7571   0.5678
    8   0.0500   0.3750   0.0000   0.6571   0.4928
    9   0.0500   0.4250   0.0000   0.5179   0.3885
   10   0.0500   0.4750   0.0000   0.2180   0.1635

    0.77612 = actual lift coefficient
    -.25871 = actual moment coefficient
    0.02847 = induced drag coefficient
    1.01005 = span efficiency factor
```

Appendix I: Matlab Utility readidrag.m

```
function [header,case_title,input_mode,write_flag,sym_flag,cl_design, ...
 cm_flag,cm_design,xcg,cp,sref,cavg,npanels,xc,yc,zc,nvortices,spacing, ...
 x,y,z,load,cn,cl_actual,cm_actual,cd_induced,e] = readidrag(filename)
%READIDRAG Reads the output file created by the code 'idrag'.
%          READIDRAG(FILENAME)
%            FILENAME - name of idrag output file (entered within single quotes)
%
% Created by Joel Grasmeyer
% Last modified on 02/03/97

fid = fopen(filename,'r');

header = fscanf(fid,'%s %s %s',3);
case_title = fscanf(fid,'%s',1);
input_mode = fscanf(fid,'%f %s %s %s',4); input_mode = input_mode(1);
write_flag = fscanf(fid,'%f %s %s %s',4); write_flag = write_flag(1);
sym_flag = fscanf(fid,'%f %s %s %s',4); sym_flag = sym_flag(1);
cl_design = fscanf(fid,'%f %s %s %s %s',5); cl_design = cl_design(1);
cm_flag = fscanf(fid,'%f %s %s %s %s %s',5); cm_flag = cm_flag(1);
cm_design = fscanf(fid,'%f %s %s %s %s %s',5); cm_design = cm_design(1);
xcg = fscanf(fid,'%f %s %s %s %s %s',5); xcg = xcg(1);
cp = fscanf(fid,'%f %s %s %s %s %s %s %s %s',8); cp = cp(1);
sref = fscanf(fid,'%f %s %s %s %s',4); sref = sref(1);
cavg = fscanf(fid,'%f %s %s %s %s',4); cavg = cavg(1);
npanels = fscanf(fid,'%f %s %s %s %s %s',5); npanels = npanels(1);
for i=1:npanels,
  junk = fscanf(fid,'%s %s %s %s %s %f',6);
  for j=1:4,
    xc(j,i) = fscanf(fid,'%f',1);
    yc(j,i) = fscanf(fid,'%f',1);
    zc(j,i) = fscanf(fid,'%f',1);
  end
  nvor = fscanf(fid,'%f %s %s %s %s',5); nvortices(i) = nvor(1);
  spac = fscanf(fid,'%f %s %s %s %s',5); spacing(i) = spac(1);
end
junk = fscanf(fid,'%s %s %s %s %s %s',6);
k = 1;
for i=1:npanels,
  for j=1:nvortices(i),
    num = fscanf(fid,'%f',1);
    x(j,i) = fscanf(fid,'%f',1);
    y(j,i) = fscanf(fid,'%f',1);
    z(j,i) = fscanf(fid,'%f',1);
    load(k) = fscanf(fid,'%f',1);
    cn(k) = fscanf(fid,'%f',1);
    k = k + 1;
  end
end
cl_actual = fscanf(fid,'%f %s %s %s %s',5); cl_actual = cl_actual(1);
cm_actual = fscanf(fid,'%f %s %s %s %s',5); cm_actual = cm_actual(1);
cd_induced = fscanf(fid,'%f %s %s %s %s',5); cd_induced = cd_induced(1);
e = fscanf(fid,'%f %s %s %s %s',5); e = e(1);

fclose(fid);
```

Appendix J: Matlab Utility geom.m

```
function geom(filename,view_flag,vortex_flag)
%GEOM Creates plots of the configuration geometry.
%     GEOM(FILENAME,VIEW_FLAG,VORTEX_FLAG)
%       FILENAME  - name of idrag output file (entered within single quotes)
%       VIEW_FLAG - flag to denote the content of the plots:
%         0 = 1 perspective plot, 1 = 1 perspective and 3-views
%       VORTEX_FLAG - flag to determine whether vortex locations are shown:
%         0 = don't plot vortex locations, 1 = show vortex locations
%
%     If GEOM is called with just the FILENAME argument, VIEW_FLAG and
%       VORTEX_FLAG are set to their default values of 1 and 0, respectively.
%
% Created by Joel Grasmeyer
% Last modified on 12/2/96

% Set default plot parameters for 1 input argument
if nargin == 1
  view_flag = 1;
  vortex_flag = 0;
end

% Read idrag output file
[header,case_title,input_mode,write_flag,sym_flag,cl_design, ...
 cm_flag,cm_design,xcg,cp,sref,cavg,npanels,xc,yc,zc,nvortices,spacing, ...
 x,y,z,loading,cn,cl_actual,cm_actual,cd_induced,e] = readidrag(filename);

% Calculate ranges for plots
minx = min(min(xc));
miny = min(min(yc));
minz = min(min(zc));
maxx = max(max(xc));
maxy = max(max(yc));
maxz = max(max(zc));
[minmin,imin] = min([minx miny minz]);
[maxmax,imax] = max([maxx maxy maxz]);
del = maxmax - minmin;
margin = 0.05;
amin = minmin - margin*del;
amax = maxmax + margin*del;

% Close panels
xc(5,:) = xc(1,:);
yc(5,:) = yc(1,:);
zc(5,:) = zc(1,:);

% Create plots, depending on view_flag
if view_flag == 0
  for i=1:npanels
    if sym_flag == 1
      plot3(yc(:,i),xc(:,i),zc(:,i),'y-',-yc(:,i),xc(:,i),zc(:,i),'y-')
      axis([-amax amax amin amax amin amax])
    else
      plot3(yc(:,i),xc(:,i),zc(:,i),'y-')
      axis([amin amax amin amax amin amax])
    end
    title(['Perspective View (e = ',num2str(e),')'])
    xlabel('y')
```

```
    ylabel('x')
    zlabel('z')
    set(gca,'YDir','rev')
    hold on
    if vortex_flag == 1
      plot3(y(1:nvortices(i),i),x(1:nvortices(i),i),z(1:nvortices(i),i),'b+')
    end
  end
  hold off
else
  for i=1:npanels
    if sym_flag == 1
      subplot(2,2,1), plot(yc(:,i),xc(:,i),'y-',-yc(:,i),xc(:,i),'y-')
      axis([-amax amax amin amax])
      axis('equal')
    else
      subplot(2,2,1), plot(yc(:,i),xc(:,i),'y-')
      axis([amin amax amin amax])
    end
    title(['Top View (e = ',num2str(e),')'])
    xlabel('y')
    ylabel('x')
    axis('equal')
    set(gca,'YDir','rev')
    hold on
    if vortex_flag == 1
      subplot(2,2,1), plot(y(1:nvortices(i),i),x(1:nvortices(i),i),'b+')
    end

    if sym_flag == 1
      subplot(2,2,2), plot3(yc(:,i),xc(:,i),zc(:,i),'y-', ...
       -yc(:,i),xc(:,i),zc(:,i),'y-')
      axis([-amax amax amin amax amin amax])
    else
      subplot(2,2,2), plot3(yc(:,i),xc(:,i),zc(:,i),'y-')
      axis([amin amax amin amax amin amax])
    end
    title('Perspective View')
    xlabel('y')
    ylabel('x')
    zlabel('z')
    set(gca,'YDir','rev')
    hold on
    if vortex_flag == 1
      subplot(2,2,2), plot3(y(1:nvortices(i),i),x(1:nvortices(i),i), ...
       z(1:nvortices(i),i),'b+')
    end

    if sym_flag == 1
      subplot(2,2,3), plot(yc(:,i),zc(:,i),'y-',-yc(:,i),zc(:,i),'y-')
      axis([-amax amax amin amax])
    else
      subplot(2,2,3), plot(yc(:,i),zc(:,i),'y-')
      axis([amin amax amin amax])
    end
    title('Rear View')
    xlabel('y')
    ylabel('z')
    axis('equal')
    hold on
    if vortex_flag == 1
```

```
      subplot(2,2,3), plot(y(1:nvortices(i),i),z(1:nvortices(i),i),'b+')
    end

    subplot(2,2,4), plot(xc(:,i),zc(:,i),'y-')
    title('Right Side View')
    xlabel('x')
    ylabel('z')
    axis([amin amax amin amax])
    axis('equal')
    set(gca,'XDir','rev')
    hold on
    if vortex_flag == 1
      subplot(2,2,4), plot(x(1:nvortices(i),i),z(1:nvortices(i),i),'b+')
    end
  end
  subplot(2,2,1),hold off
  subplot(2,2,2),hold off
  subplot(2,2,3),hold off
  subplot(2,2,4),hold off
end
```

Appendix K: Matlab Utility loads.m

```
function loads(filename,dist_flag)
%LOADS Creats plots of load or normal force coefficient distribution.
%      LOADS(FILENAME,DIST_FLAG)
%        FILENAME  - name of idrag output file (entered within single quotes)
%        DIST_FLAG - flag to denote the content of the plots:
%         (0 = loading (cn*c/cavg), 1 = normal force coefficient (cn))
%
%     If LOADS is called with just the FILENAME argument, DIST_FLAG is
%       set to its default value of 0.
%
% Created by Joel Grasmeyer
% Last modified on 12/2/96

% Set default plot parameters for 1 input argument
if nargin == 1
  dist_flag = 0;
end

% Read idrag output file
[header,case_title,input_mode,write_flag,sym_flag,cl_design, ...
 cm_flag,cm_design,xcg,cp,sref,cavg,npanels,xc,yc,zc,nvortices,spacing, ...
 x,y,z,loading,cn,cl_actual,cm_actual,cd_induced,e] = readidrag(filename);

% Set distribution to cn or load, depending on dist_flag
if dist_flag == 0
  dist = loading;
  label = 'Load Distribution (Cn*c/cavg)';
else
  dist = cn;
  label = 'Normal Force Coefficient Distribution (Cn)';
end
xmin = 0;
xmax = max( sqrt( (yc(2,:)-yc(1,:)).^2 + (zc(2,:)-zc(1,:)).^2 ) );
ymin = min(dist);
ymax = max(dist);

% Determine number of subplots to create (maximum of 4)
if npanels == 1
  rows = 1;
  cols = 1;
elseif npanels == 2
  rows = 2;
  cols = 1;
elseif npanels == 3 | npanels == 4
  rows = 2;
  cols = 2;
end

% Plot cn or load distribution for each panel
if npanels >= 1
  station = sqrt( (y(1:nvortices(1),1) - yc(1,1)).^2 + ...
   (z(1:nvortices(1),1) - zc(1,1)).^2 );
  subplot(rows,cols,1), plot(station,dist(1:nvortices(1)),'-')
  title(['Panel 1 (e = ',num2str(e),')'])
  ylabel(label)
  xlabel('Distance Along Panel')
  axis([xmin xmax ymin ymax])
```

```
  grid on
end
if npanels >= 2
  station = sqrt( (y(1:nvortices(2),2) - yc(1,2)).^2 + ...
   (z(1:nvortices(2),2) - zc(1,2)).^2 );
  subplot(rows,cols,2), plot(station,dist(nvortices(1)+1:sum(nvortices(1:2))),'-')
  title('Panel 2')
  xlabel('Distance Along Panel')
  ylabel(label)
  axis([xmin xmax ymin ymax])
  grid on
end
if npanels >= 3
  station = sqrt( (y(1:nvortices(3),3) - yc(1,3)).^2 + ...
   (z(1:nvortices(3),3) - zc(1,3)).^2 );
  subplot(rows,cols,3), plot(station, ...
   dist(sum(nvortices(1:2))+1:sum(nvortices(1:3))),'-')
  title('Panel 3')
  xlabel('Distance Along Panel')
  ylabel(label)
  axis([xmin xmax ymin ymax])
  grid on
end
if npanels >= 4
  station = sqrt( (y(1:nvortices(4),4) - yc(1,4)).^2 + ...
   (z(1:nvortices(4),4) - zc(1,4)).^2 );
  subplot(rows,cols,4), plot(station, ...
   dist(sum(nvortices(1:3))+1:sum(nvortices(1:4))),'-')
  title('Panel 4')
  xlabel('Distance Along Panel')
  ylabel(label)
  axis([xmin xmax ymin ymax])
  grid on
end
```