AVL Official guide

Stesura a cura di
Bruno Spoti

AVL 3.36 User Primer last update 12 Feb 17 Mark Drela, MIT Aero & Astro Harold Youngren, Aerocraft, Inc.

Indice

Capitolo 1

History and General description

AVL (Athena Vortex Lattice) 1.0 was originally written by Harold Youngren circa 1988 for the MIT Athena TODOR aero software collection. The code was based on classic work by Lamar (NASA codes), E. Lan and L. Miranda (VORLAX) and a host of other investigators. Numerous modifications have since been added by Mark Drela and Harold Youngren, to the point where only stubborn traces of the original Athena code remain.

1.1 General Description

AVL 3.xx now has a large number of features intended for rapid aircraft configuration analysis. The major features are as follows:

Aerodynamic components

- Lifting surfaces
- Slender bodies

Configuration description

- Keyword-driven geometry input file
- Defined sections with linear interpolation
- Section properties
  - camberline is NACA xxxx, or from airfoil file
  - control deflections
  - parabolic profile drag polar, Re-scaling
- Scaling, translation, rotation of entire surface or body
- Duplication of entire surface or body

Singularities

- Horseshoe vortices (surfaces)
- Source+doublet lines (bodies)
- Finite-core option

Discretization

- Uniform
- Sine
- Cosine
- Blend

Control deflections

- Via normal-vector tilting
- Leading edge flaps
- Trailing edge flaps
- Hinge lines independent of discretization

General freestream description

- alpha,beta flow angles
- p,q,r aircraft rotation components
- Subsonic Prandtl-Glauert compressibility treatment

Surfaces can be defined to "see" only perturbation velocities (not freestream) to allow simulation of:

- ground effect
- wind tunnel wall interference
- influence of other nearby aircraft

Aerodynamic outputs

- Direct forces and moments
- Trefftz-plane
- Derivatives of forces and moments, w.r.t freestream, rotation, controls
- In body or stability axes

Trim calculation

- Operating variables
  - alpha,beta
  - p,q,r
  - control deflections
- Constraints
  - direct constraints on variables
  - indirect constraints via specified CL, moments
- Multiple trim run cases can be defined
- Saving of trim run case setups for later recall

Optional mass definition file (only for trim setup, eigenmode analysis)

- User-chosen units
- Itemized component location, mass, inertias

Trim setup of constraints

- level or banked horizontal flight
- steady pitch rate (looping) flight

Eigenmode analysis

- Rigid-body analysis with quasi-steady aero model
- Display of eigenvalue root progression with a parameter
- Display of eigenmode motion in real time
- Output of dynamic system matrices

Capitolo 2

Vortex-Lattice Modeling Principles

Like any computational method, AVL has limitations on what it can do. These must be kept in mind in any given application.

2.1 Configurations

A vortex-lattice model like AVL is best suited for aerodynamic configurations which consist mainly of thin lifting surfaces at small angles of attack and sideslip. These surfaces and their trailing wakes are represented as single-layer vortex sheets, discretized into horseshoe vortex filaments, whose trailing legs are assumed to be parallel to the x-axis. AVL provides the capability to also model slender bodies such as fuselages and nacelles via source+doublet filaments. The resulting force and moment predictions are consistent with slender-body theory, but the experience with this model is relatively limited, and hence modeling of bodies should be done with caution. If a fuselage is expected to have little influence on the aerodynamic loads, it's simplest to just leave it out of the AVL model. However, the two wings should be connected by a fictitious wing portion which spans the omitted fuselage.

2.2 Unsteady flow

AVL assumes quasi-steady flow, meaning that unsteady vorticity shedding is neglected. More precisely, it assumes the limit of small reduced frequency, which means that any oscillatory motion (e.g. in pitch) must be slow enough so that the period of oscillation is much longer than the time it takes the flow to traverse an airfoil chord. This is true for virtually any expected flight maneuver. Also, the roll, pitch, and yaw rates used in the computations must be slow enough so that the resulting relative flow angles are small. This can be judged by the dimensionless rotation rate parameters, which should fall within the following practical limits.

$$-0.10 < \frac{pb}{2V} < 0.10 \qquad -0.03 < \frac{qc}{2V} < 0.03 \qquad -0.25 < \frac{rb}{2V} < 0.25$$

These limits represent extremely violent aircraft motion, and are unlikely to exceeded in any typical flight situation, except possibly during low-airspeed aerobatic maneuvers. In any case, if any of these parameters falls outside of these limits, the results should be interpreted with caution.

2.3 Compressibility

Compressibility is treated in AVL using the classical Prandtl-Glauert (PG) transformation, which converts the PG equation to the Laplace equation, which can then be solved by the basic incompressible method. This is equivalent to the compressible continuity equation, with the assumptions of irrotationality and linearization about the freestream. The forces are computed by applying the Kutta-Joukowsky relation to each vortex, this remaining valid for compressible flow.

The linearization assumes small perturbations (thin surfaces) and is not completely valid when velocity perturbations from the free-stream become large. The relative importance of compressible effects can be judged by the PG factor $\frac{1}{B} = \frac{1}{\sqrt{1-M^2}}$, where "$M$" is the freestream Mach number. A few values are given in the table, which shows the expected range of validity.

M	$\frac{1}{B}$	PG validity
0.0	1.000	
0.1	1.005	
0.2	1.021	
0.3	1.048	PG expected valid
0.4	1.091	
0.5	1.155	
0.6	1.250	
0.7	1.400	PG suspect (transonic flow likely)
0.8	1.667	PG unreliable (transonic flow certain)
0.9	2.294	PG hopeless

Tabella 2.1. Prandtl-Glauert expected range of validity

For swept-wing configurations, the validity of the PG model is best judged using the wing-perpendicular Mach number

$$M_\perp = M\cos(\Lambda)$$

Since Mperp $<$ M, swept-wing cases can be modeled up to higher M values than unswept cases. For example, a 45 degree swept wing operating at freestream M = 0.8 has

$$M_\perp = 0.8\cos(45^\circ) = 0.566$$

which is still within the expected range of PG validity in the above table. So reasonable results can be expected from AVL for this case.

When doing velocity parameter sweeps at the lowest Mach numbers, say below $M = 0.2$, it is best to simply hold $M = 0$. This will greatly speed up the calculations, since changing the Mach number requires recomputation and re-factorization of the VL influence matrix, which consumes most of the computational effort. If the Mach number is held fixed, this computation needs to be done only once.

Capitolo 3

Input Files

AVL works with three input files, all in plain text format. Ideally these all have a common arbitrary prefix "xxx", and the following extensions:

- xxx.avl required main input file defining the configuration geometry
- xxx.mass optional file giving masses and inertias, and dimensional units
- xxx.run optional file defining parameters for some number of run cases

The user provides files xxx.avl and xxx.mass, which are typically created using any text editor. Sample files are provided for use as templates. The xxx.run file is written by AVL itself with a user command. It can be manually edited, although this is not really necessary since it is more convenient to edit the contents in AVL and then write out the file again.

3.1 Geometry Input File – xxx.avl

This file describes the vortex lattice geometry and aerodynamic section properties. Sample input files are in the runs/ subdirectory.

3.1.1 Coordinate system

The geometry is described in the following Cartesian system:
X downstream Y out the right wing Z up

The freestream must be at a reasonably small angle to the X axis (alpha and beta must be small), since the trailing vorticity is oriented parallel to the X axis. The length unit used in this file is referred to as "Lunit". This is arbitrary, but must be the same throughout this file.

3.1.2 File format

Header data

The input file begins with the following information in the first 5 non-blank, non-comment lines:

```
Abc...                | case title

#                     | comment line begins with "#" or "!"

0.0                   | Mach
1     0     0.0       | iYsym  iZsym  Zsym
4.0   0.4   0.1       | Sref   Cref   Bref
0.1   0.0   0.0       | Xref   Yref   Zref
0.020                 | CDp  (optional)
```

- Mach = default freestream Mach number for Prandtl-Glauert correction
- iYsym = 1 case is symmetric about Y=0 , (X-Z plane is a solid wall)
  iYsym = -1 case is antisymmetric about Y=0, (X-Z plane is at const. Cp)
  iYsym = 0 no Y-symmetry is assumed
- iZsym = 1 case is symmetric about Z=Zsym , (X-Y plane is a solid wall)
  iZsym = -1 case is antisymmetric about Z=Zsym, (X-Y plane is at const. Cp)
  iZsym = 0 no Z-symmetry is assumed (Zsym ignored)
- Sref = reference area used to define all coefficients (CL, CD, Cm, etc)
- Cref = reference chord used to define pitching moment (Cm)
- Bref = reference span used to define roll,yaw moments (Cl,Cn)
- X,Y,Zref = default location about which moments and rotation rates are defined (if doing trim calculations, XYZref must be the CG location, which can be imposed with the MSET command described later)
- CDp = default profile drag coefficient added to geometry, applied at XYZref (assumed zero if this line is absent, for previous-version compatibility)

The default Mach, XYZref, and CDp values are superseded by the values in the .run file (described later), if it is present. They can also be changed at runtime.

Only the half (non-image) geometry must be input if symmetry is specified. Ground effect is simulated with iZsym = 1, and Zsym = location of ground.

Forces are not calculated on the image/anti-image surfaces. Sref and Bref are assumed to correspond to the total geometry.

In practice there is little reason to run Y-symmetric image cases, unless one is desperate for CPU savings.

Surface and Body data

The remainder of the file consists of a set of keywords and associated data. Each keyword expects a certain number of lines of data to immediately follow it, the exception being inline-coordinate keyword AIRFOIL which is followed by an arbitrary number of coordinate data lines. The keywords must also be nested properly in the hierarchy shown below. Only the first four characters of each keyword are actually significant, the rest are just a mnemonic.

```
SURFACE
  COMPONENT (or INDEX)
  YDUPLICATE
  SCALE
  TRANSLATE
  ANGLE
  NOWAKE
  NOALBE
  NOLOAD
  CDCL

SECTION

SECTION
  NACA

SECTION
  AIRFOIL
  CLAF
  CDCL

SECTION
  AFILE
  CONTROL
  CONTROL

BODY
  YDUPLICATE
  SCALE
  TRANSLATE
  BFILE


SURFACE
  YDUPLICATE

  SECTION
```

```
  SECTION

SURFACE
 .
 .
 etc.
```

The COMPONENT (or INDEX), YDUPLICATE, SCALE, TRANSLATE, and ANGLE keywords can all be used together. If more than one of these appears for a surface, the last one will be used and the previous ones ignored.

At least two SECTION keywords must be used for each surface.

The NACA, AIRFOIL, AFILE, keywords are alternatives. If more than one of these appears after a SECTION keyword, the last one will be used and the previous ones ignored. i.e.

SECTION NACA AFILE

is equivalent to

SECTION AFILE

Multiple CONTROL keywords can appear after a SECTION keyword and data

Surface-definition keywords and data formats

```
  *****

  SURFACE                | (keyword)
  Main Wing              | surface name string
  12   1.0  20  -1.5     | Nchord  Cspace   [ Nspan Sspace ]
```

The SURFACE keyword declares that a surface is being defined until the next SURFACE or BODY keyword, or the end of file is reached. A surface does not really have any significance to the underlying AVL vortex lattice solver, which only recognizes the overall collection of all the individual horseshoe vortices. SURFACE is provided only as a configuration-defining device, and also as a means of defining individual surface forces. This is necessary for structural load calculations, for example.

- Nchord = number of chordwise horseshoe vortices placed on the surface
- Cspace = chordwise vortex spacing parameter (described later)
- Nspan = number of spanwise horseshoe vortices placed on the surface [optional]
- Sspace = spanwise vortex spacing parameter (described later) [optional]

If Nspan and Sspace are omitted (i.e. only Nchord and Cspace are present on line), then the Nspan and Sspace parameters will be expected for each section interval, as described later.

*****

```
COMPONENT        | (keyword) or INDEX
3                | Lcomp
```

This optional keywords COMPONENT (or INDEX for backward compatibility) allows multiple input SURFACEs to be grouped together into a composite virtual surface, by assigning each of the constituent surfaces the same Lcomp value. Application examples are:

- A wing component made up of a wing SURFACE and a winglet SURFACE
- A T-tail component made up of horizontal and vertical tail SURFACEs.

A common Lcomp value instructs AVL to *not* use a finite-core model for the influence of a horseshoe vortex and a control point which lies on the same component, as this would seriously corrupt the calculation.

If each COMPONENT is specified via only a single SURFACE block, then the COMPONENT (or INDEX) declaration is unnecessary.

```
*****
```

```
YDUPLICATE       | (keyword)
0.0              | Ydupl
```

The YDUPLICATE keyword is a convenient shorthand device for creating another surface which is a geometric mirror image of the one being defined. The duplicated surface is *not* assumed to be an aerodynamic image or anti-image, but is truly independent. A typical application would be for cases which have geometric symmetry, but not aerodynamic symmetry, such as a wing in yaw. Defining the right wing together with YDUPLICATE will conveniently create the entire wing.

The YDUPLICATE keyword can *only* be used if iYsym = 0 is specified. Otherwise, the duplicated real surface will be identical to the implied aerodynamic image surface, and velocities will be computed directly on the line-vortex segments of the images. This will almost certainly produce an arithmetic fault.

The duplicated surface gets the same Lcomp value as the parent surface, so they are considered to be the same COMPONENT. There is no significant effect on the results if they are in reality two physically-separate surfaces.

Ydupl = Y position of X-Z plane about which the current surface is reflected to make the duplicate geometric-image surface.

```
*****
```

```
SCALE            |  (keyword)
1.0  1.0  0.8    | Xscale  Yscale  Zscale
```

The SCALE allows convenient rescaling for the entire surface. The scaling is applied before the TRANSLATE operation described below.

Xscale,Yscale,Zscale = scaling factors applied to all x,y,z coordinates (chords are also scaled by Xscale)

```
*****

TRANSLATE          |  (keyword)
10.0  0.0  0.5     | dX  dY  dZ
```

The TRANSLATE keyword allows convenient relocation of the entire surface without the need to change the Xle,Yle,Zle locations for all the defining sections. A body can be translated without the need to modify the body shape coordinates.

dX,dY,dZ = offset added on to all X,Y,Z values in this surface.

```
*****

ANGLE        |  (keyword)
2.0          | dAinc
```

The ANGLE keyword allows convenient changing of the incidence angle of the entire surface without the need to change the Ainc values for all the defining sections.

dAinc = offset added on to the Ainc values for all the defining sections in this surface

```
*****

NOWAKE     |  (keyword)
```

The NOWAKE keyword specifies that this surface is to NOT shed a wake, so that its strips will not have their Kutta conditions imposed. Such a surface will have a near-zero net lift, but it will still generate a nonzero moment.

```
*****

NOALBE     |  (keyword)
```

The NOALBE keyword specifies that this surface is unaffected by freestream direction changes specified by the alpha,beta angles and p,q,r rotation rates. This surface then reacts to only to the perturbation velocities of all the horseshoe vortices and sources and doublets in the flow. This allows the SURFACE/NOALBE object to model fixed surfaces such as a ground plane, wind tunnel walls, or a nearby other aircraft which is at a fixed flight condition.

```
*****

NOLOAD     |  (keyword)
```

The NOLOAD keyword specifies that the force and moment on this surface is to NOT be included in the overall forces and moments of the configuration. This is typically used

together with NOALBE, since the force on a ground plane or wind tunnel walls certainly is not to be considered as part of the aircraft force of interest.

The following keyword declarations would be used in envisioned applications.

1. Non-lifting fuselage modeled by its side-view and top-view profiles. This will capture the moment of the fuselage reasonably well.

   ```
   NOWAKE
   ```

2. Another nearby aircraft, with both aircraft maneuvering together. This would be for trim calculation in formation flight.

   ```
   NOALBE
   NOLOAD
   ```

3. Another nearby aircraft, with only the primary aircraft maneuvering. This would be for a flight-dynamics analysis in formation flight.

   ```
   NOLOAD
   ```

4. Nearby wind tunnel walls or ground plane.

   ```
   NOALBE
   NOLOAD
   ```

```
*****

CDCL                          |  (keyword)
CL1 CD1  CL2 CD2  CL3 CD3     |  CD(CL) function parameters
```

The CDCL keyword placed in the SURFACE options specifies a simple profile-drag CD(CL) function for all sections in this SURFACE. The function is parabolic between CL1..CL2 and CL2..CL3, with rapid increases in CD below CL1 and above CL3.

The CD-CL polar is based on a simple interpolation with four CL regions:

1. negative stall region
2. parabolic CD(CL) region between negative stall and the drag minimum
3. parabolic CD(CL) region between the drag minimum and positive stall
4. positive stall region

```
CLpos,CDpos       <-  Region 4 (quadratic above CLpos)
CL |   pt3--------
|    /
|   |                    <-  Region 3 (quadratic above CLcdmin)
| pt2 CLcdmin,CDmin
|   |
|    \                   <-  Region 2 (quadratic below CLcdmin)
```

```
|   pt1_________
|     CLneg,CDneg       <-  Region 1 (quadratic below CLneg)
|
-------------------------
CD
```

See the SUBROUTINE CDCL header (in cdcl.f) for more details.

The CD(CL) function is interpolated for stations in between defining sections.

```
*****
```

```
SECTION                                 |  (keyword)
0.0 5.0 0.2   0.50  1.50   5 -2.0   | Xle Yle Zle   Chord Ainc   [ Nspan Sspace ]
```

The SECTION keyword defines an airfoil-section camber line at some spanwise location on the surface.

- Xle,Yle,Zle = airfoil's leading edge location
- Chord = the airfoil's chord (trailing edge is at Xle+Chord,Yle,Zle)
- Ainc = incidence angle, taken as a rotation (+ by RH rule) about the surface's spanwise axis projected onto the Y-Z plane.
- Nspan = number of spanwise vortices until the next section [optional]
- Sspace = controls the spanwise spacing of the vortices [optional]

Nspan and Sspace are used here only if the overall Nspan and Sspace for the whole surface is not specified after the SURFACE keyword. The Nspan and Sspace for the last section in the surface are always ignored.

Note that Ainc is used only to modify the flow tangency boundary condition on the airfoil camber line, and does not rotate the geometry of the airfoil section itself. This approximation is consistent with linearized airfoil theory.

The local chord and incidence angle are linearly interpolated between defining sections. Obviously, at least two sections (root and tip) must be specified for each surface.

The default airfoil camber line shape is a flat plate. The NACA, AIRFOIL, and AFIL keywords, described below, are available to define non-flat camber lines. If one of these is used, it must immediately follow the data line of the SECTION keyword.

A positive surface-airfoil circulation and a corresponding positive local lift coefficient is defined by righthand rule in direction of successive sections. This also defines the tops of the section airfoils as the suction side for positive overall lift. Therefore, to match conventional definitions, the order of the sections must be left to right across the span.

NOTE!! If the sections are ordered right to left, then the overall airfoils will effectively by upside down. The overall dCL/dalpha will still be positive as usual, but for positive CL the local cl values will be negative. Needless to say, it's best to avoid these complications by ordering the sections left to right across the span (root to tip for the right wing).

The section ordering of a vertical tail is somewhat more arbitrary. But a top to bottom ordering is most convenient, since positive local cl values then produce a positive yaw moment Cn.

```
*****

NACA      X1  X2          | (keyword)    [ optional x/c range ]
4300                      | section NACA camberline
```

The NACA keyword sets the camber line to the NACA 4-digit shape specified.

If present, the optional X1 X2 numerical parameters indicate that only the x/c range X1..X2 from the coordinates is to be assigned to the surface. If the surface is a 20%-chord flap, for example, then X1 X2 would be 0.80 1.00. This allows the camber shape to be easily assigned to any number of surfaces in piecewise manner.

If omitted, X1, X2 default to 0.0, 1.0, which indicates that the entire airfoil is to be used to define the camber shape as usual.

```
*****

AIRFOIL   X1  X2          | (keyword)    [ optional x/c range ]
1.0   0.0                 | x/c(1)  y/c(1)
0.98  0.002               | x/c(2)  y/c(2)
.     .                   |  .       .
.     .                   |  .       .
.     .                   |  .       .
1.0  -0.01                | x/c(N)  y/c(N)
```

The AIRFOIL keyword declares that the airfoil definition is input as a set of x/c, y/c pairs.

x/c,y/c = airfoil coordinates

The x/c, y/c coordinates run from TE, to LE, back to the TE again in either direction. These coordinates are splined, and the slope of the camber y(x) function is obtained from the middle y/c values between the top and bottom. The number of points N is deterimined when a line without two readable numbers is encountered.

If present, the optional X1 X2 parameters indicate that only the x/c range X1..X2 from the coordinates is to be assigned to the surface. If the surface is a 20%-chord flap, for example, then X1 X2 would be 0.80 1.00. This allows the camber shape to be easily assigned to any number of surfaces in piecewise manner.

```
*****

AFILE      X1  X2         | (keyword)   [ optional x/c range ]
filename                  | filename string
```

The AFILE keyword is essentially the same as AIRFOIL, except that the x/c,y/c pairs are generated from a standard (XFOIL-type) set of airfoil coordinates contained in the

file "filename". The first line of this file is assumed to contain a string with the name of the airfoil (as written out with XFOIL's SAVE command). If the path/filename has embedded blanks, double quotes should be used to delimit the string.

If present, the optional X1 X2 numerical parameters indicate that only the x/c range X1..X2 from the coordinates is to be assigned to the surface. If the surface is a 20%-chord flap, for example, then X1 X2 would be 0.80 1.00. This allows the camber shape to be easily assigned to any number of surfaces in piecewise manner.

If omitted, X1, X2 default to 0.0, 1.0, which indicates that the entire airfoil is to be used to define the camber shape as usual.

```
*****

DESIGN                    | (keyword)
DName  Wdes               | design parameter name,  local weight
```

This declares that the section angle Ainc is to be virtually perturbed by a design parameter, with name DName and local weight Wdes.

For example, declarations for design variables "twist1" and "bias1"

```
DESIGN
twist1  -0.5

DESIGN
bias1   1.0
```

Give an effective (virtual) section incidence that is set using the "twist1" and "bias1" design variables as:

```
Ainc_total = Ainc  - 0.5*twist1_value + 1.0*bias_value
```

where `twist1_value` and `bias1_value` are design parameters specified at runtime.

The sensitivities of the flow solution to design variable changes can be displayed at any time during program execution. Hence, design variables can be used to quickly investigate the effects of twist changes on lift, moments, induced drag, etc.

Declaring the same design parameter with varying weights for multiple sections in a surface allows the design parameter to represent a convenient "design mode", such as linear washout, which influences all sections.

```
*****

CONTROL                          | (keyword)
elevator  1  0.6  0  1  0  1     | name, gain,  Xhinge,  XYZhvec,  SgnDup
```

The CONTROL keyword declares that a hinge deflection at this section is to be governed by one or more control variables. An arbitrary number of control variables can be used, limited only by the array limit NDMAX.

The data line quantities are...

- name = name of control variable

- gain = control deflection gain, units: degrees deflection / control variable
- Xhinge = x/c location of hinge.
  - If positive, control surface extent is Xhinge..1 (TE surface)
  - If negative, control surface extent is 0..-Xhinge (LE surface)
- XYZhvec = vector giving hinge axis about which surface rotates + deflection is + rotation about hinge vector by righthand rule Specifying XYZhvec = 0. 0. 0. puts the hinge vector along the hinge
- SgnDup = sign of deflection for duplicated surface
  - An elevator would have SgnDup = +1
  - An aileron would have SgnDup = -1

Control derivatives will be generated for all control variables which are declared.

More than one variable can contribute to the motion at a section. For example, for the successive declarations

```
CONTROL
aileron  1.0  0.7  0. 1. 0.  -1.0

CONTROL
flap     0.3  0.7  0. 1. 0.   1.0
```

the overall deflection will be

```
control_surface_deflection  =  1.0 * aileron  +  0.3 * flap
```

The same control variable can be used on more than one surface. For example the wing sections might have

```
CONTROL
flap     0.3   0.7  0. 1. 0.   1.0
```

and the horizontal tail sections might have

```
CONTROL
flap     0.03  0.5  0. 1. 0.   1.0
```

with the latter simulating 10:1 flap -> elevator mixing.

A partial-span control surface is specified by declaring CONTROL data only at the sections where the control surface exists, including the two end sections. For example, the following wing defined with three sections (i.e. two panels) has a flap over the inner panel, and an aileron over the outer panel.

```
SECTION
0.0  0.0  0.0   2.0   0.0   | Xle Yle Zle   Chord Ainc
CONTROL
flap     1.0   0.80   0. 0. 0.   1   | name, gain,  Xhinge,  XYZhvec,
SgnDup

SECTION
0.0  8.0  0.0   2.0   0.0   | Xle Yle Zle   Chord Ainc
CONTROL
flap     1.0   0.80   0. 0. 0.   1   | name, gain,  Xhinge,  XYZhvec,
SgnDup
CONTROL
aileron  1.0   0.85   0. 0. 0.  -1   | name, gain,  Xhinge,  XYZhvec,
SgnDup

SECTION
0.2 12.0  0.0   1.5   0.0   | Xle Yle Zle   Chord Ainc
CONTROL
aileron  1.0   0.85   0. 0. 0.  -1   | name, gain,  Xhinge,  XYZhvec,
SgnDup
```

The control gain for a control surface does not need to be equal at each section. Spanwise stations between sections receive a gain which is linearly interpolated from the two bounding sections. This allows specification of flexible-surface control systems. For example, the following surface definition models wing warping which is linear from root to tip. Note that the "hinge" is at x/c=0.0, so that the entire chord rotates in response to the aileron deflection.

```
SECTION
0.0  0.0  0.0   2.0   0.0   | Xle Yle Zle   Chord Ainc
CONTROL
aileron  0.0   0.     0. 0. 0.  -1   | name, gain,  Xhinge,  XYZhvec,
SgnDup

SECTION
0.2 12.0  0.0   1.5   0.0   | Xle Yle Zle   Chord Ainc
CONTROL
aileron  1.0   0.     0. 0. 0.  -1   | name, gain,  Xhinge,  XYZhvec,
SgnDup
```

Non-symmetric control effects, such as Aileron Differential, can be specified by a non-unity SgnDup magnitude. For example,

```
SECTION
0.0  6.0  0.0   2.0   0.0   | Xle Yle Zle   Chord Ainc
CONTROL
```

```
aileron  1.0   0.7    0. 0. 0.  -2.0   | name, gain,  Xhinge,  XYZhvec,
SgnDup

SECTION
0.0 10.0  0.0   2.0   0.0   | Xle Yle Zle   Chord Ainc
CONTROL
aileron  1.0   0.7    0. 0. 0.  -2.0   | name, gain,  Xhinge,  XYZhvec,
SgnDup
```

will result in the duplicated aileron having a deflection opposite and 2.0 times larger than the defined aileron. Note that this will have the proper effect only in one direction. In the example above, the two aileron surfaces deflect as follows:

```
Right control surface:   1.0*aileron         =  1.0*aileron
Left  control surface:   1.0*aileron*(-2.0)  = -2.0*aileron
```

which is the usual way Aileron Differential is implemented if "aileron" is positive. To get the same effect with a negative "aileron" control change, the definitions would have to be as follows.

```
SECTION
0.0  6.0  0.0   2.0   0.0   | Xle Yle Zle   Chord Ainc
CONTROL
aileron  2.0   0.7    0. 0. 0.  -0.5   | name, gain,  Xhinge,  XYZhvec,
SgnDup

SECTION
0.0 10.0  0.0   2.0   0.0   | Xle Yle Zle   Chord Ainc
CONTROL
aileron  2.0   0.7    0. 0. 0.  -0.5   | name, gain,  Xhinge,  XYZhvec,
SgnDup
```

This then gives:

```
Right control surface:   2.0*aileron         = -2.0*(-aileron)
Left  control surface:   2.0*aileron*(-0.5)  =  1.0*(-aileron)
```

which is the correct mirror image of the previous case if "aileron" is negative.

```
*****

CLAF        |  (keyword)
CLaf        |  dCL/da scaling factor
```

This scales the effective dcl/da of the section airfoil as follows:

```
dcl/da  =  2 pi CLaf
```

The implementation is simply a chordwise shift of the control point relative to the bound vortex on each vortex element.

The intent is to better represent the lift characteristics of thick airfoils, which typically have greater dcl/da values than thin airfoils. A good estimate for CLaf from 2D potential flow theory is

```
CLaf  =  1 + 0.77 t/c
```

where t/c is the airfoil's thickness/chord ratio. In practice, viscous effects will reduce the 0.77 factor to something less. Wind tunnel airfoil data or viscous airfoil calculations should be consulted before choosing a suitable CLaf value.

If the CLAF keyword is absent for a section, CLaf defaults to 1.0, giving the usual thin-airfoil lift slope dcl/da = 2 pi.

```
*****

CDCL                          |  (keyword)
CL1 CD1  CL2 CD2  CL3 CD3     |  CD(CL) function parameters
```

The CDCL keyword specifies a simple profile-drag CD(CL) function for this section. The function is parabolic between CL1..CL2 and CL2..CL3, with rapid increases in CD below CL1 and above CL3. See the SUBROUTINE CDCL header (in cdcl.f) for more details.

The CD-CL polar is based on a simple interpolation with four CL regions:

1. negative stall region
2. parabolic CD(CL) region between negative stall and the drag minimum
3. parabolic CD(CL) region between the drag minimum and positive stall
4. positive stall region

```
CLpos,CDpos       <-  Region 4 (quadratic above CLpos)
CL |   pt3--------
|    /
|   |                   <-  Region 3 (quadratic above CLcdmin)
| pt2 CLcdmin,CDmin
|   |
|    \                  <-  Region 2 (quadratic below CLcdmin)
|   pt1_________
|     CLneg,CDneg       <-  Region 1 (quadratic below CLneg)
|
-------------------------
CD
```

The CD(CL) function is interpolated for stations in between defining sections. Hence, the CDCL declaration on any surface must be used either for all sections or for none (unless the SURFACE CDCL is specified).

Body-definition keywords and data formats

```
*****

BODY                  | (keyword)
Fuselage              | body name string
15   1.0              | Nbody  Bspace
```

The BODY keyword declares that a body is being defined until the next SURFACE or BODY keyword, or the end of file is reached. A body is modeled with a source+doublet line along its axis, in accordance with slender-body theory.

- Nbody = number of source-line nodes
- Bspace = lengthwise node spacing parameter (described later)

```
*****

YDUPLICATE       | (keyword)
0.0              | Ydupl
```

Same function as for a surface, described earlier.

```
*****

SCALE            |  (keyword)
1.0  1.0  0.8    | Xscale  Yscale  Zscale
```

Same function as for a surface, described earlier.

```
*****

TRANSLATE         |  (keyword)
10.0  0.0  0.5    | dX  dY  dZ
```

Same function as for a surface, described earlier.

```
*****

BFILE    X1  X2  | (keyword)   [ optional x/c range ]
filename         | filename string
```

This specifies the shape of the body as an "airfoil" file which gives the top or side view of the body, which is assumed to have a round cross-section. Hence, the diameter of the body is the difference between the top and bottom Y values. Bodies which are not round must be approximated with an equivalent round body which has roughly the same cross-sectional areas. If the path/filename has embedded blanks double quotes should be used to delimit the string.

If present, the optional X1 X2 numerical parameters indicate that only the x/c range X1..X2 from the coordinates is to be assigned to the surface. If the body is to be defined using only the 0-80% chord points, for example, then X1 X2 would be 0.0 0.80. This allows the body shape to be truncated if needed.

If omitted, X1, X2 default to 0.0, 1.0, which indicates that the entire "airfoil" is used to define the shape as usual.

Vortex Lattice Spacing Distributions Discretization of the geometry into vortex lattice panels is controlled by the spacing parameters described earlier: Sspace, Cspace, Bspace

These must fall in the range $[-3.0, +3.0]$, and they determine the spanwise and lengthwise horseshoe vortex or body line node distributions as shown in table 3.1:

Parameter		Spacing
3.0	equal	| | | | | | | | |
2.0	sine	|| | | | | | | |
1.0	cosine	|| | | | | | ||
0.0	equal	| | | | | | | | |
-1.0	cosine	|| | | | | | ||
-2.0	-sine	| | | | | | | ||
-3.0	equal	| | | | | | | | |

Tabella 3.1. Spacing distributions parameter

- Sspace (spanwise) : first section $\Longrightarrow$ last section
- Cspace (chordwise) : leading edge $\Longrightarrow$ trailing edge
- Bspace (lengthwise): frontmost point $\Longrightarrow$ rearmost point

An intermediate parameter value will result in a blended distribution.

The most efficient distribution (best accuracy for a given number of vortices) is usually the cosine (1.0) chordwise and spanwise. If the wing does not have a significant chord slope discontinuity at the centerline, such as a straight, elliptical, or slightly tapered wing, then the -sine (-2.0) distribution from root to tip will be more efficient. This is equivalent to a cosine distribution across the whole span. The basic rule is that a tight chordwise distribution is needed at the leading and trailing edges, and a tight spanwise distribution is needed wherever the circulation is changing rapidly, such as taper breaks, and especially at flap breaks and wingtips.

The tables 3.2 and 3.2 show the accuracy superiority of the cosine spacing over uniform spacing, at least for a simple wing planform, in particular for a rectangular wing. With cosine spacing, a much smaller number of vortex elements is needed to reach the desired limiting answer to within a given tolerance. Note also that the uniform spacing always tends to overpredict the span efficiency, and its error decreases only linearly with the number of elements in each direction.

A number of vortex-spacing rules must be followed to get good results from AVL, or any other vortex-lattice method:

1. In a standard VL method, a trailing vortex leg must not pass close to a downstream control point, else the solution will be garbage. In practice, this means that surfaces

1/Ni	*Ni*	*Nj*	*CL*	*CDi*	*CLff*	*CDiff*	*e*	*e_ error*
1	1	4	4.18875	0.05807	4.19383	0.05829	0.9605	+0.09%
0.5	2	8	4.20951	0.05872	4.21465	0.05893	0.9595	−0.01%
0.25	4	16	4.21151	0.05876	4.21665	0.05898	0.9596	−0.00%
0.125	8	32	4.21184	0.05835	4.21695	0.05899	0.9596	0.00%

Tabella 3.2. Cosine spacing in C and S

1/Ni	*Ni*	*Nj*	*CL*	*CDi*	*CLff*	*CDiff*	*e*	*e_ error*
1	1	4	4.45637	0.05797	4.46144	0.05819	1.0887	+13.45%
0.5	2	8	4.35198	0.05894	4.35713	0.05917	1.0213	+6.43%
0.25	4	16	4.28694	0.05903	4.29211	0.05926	0.9896	+3.13%
0.125	8	32	4.25067	0.05895	4.25583	0.05917	0.9744	+1.54%

Tabella 3.3. Uniform spacing in C and S

which are lined up along the x direction (i.e. have the same or nearly the same y,z coordinates), MUST have the same spanwise vortex spacing. AVL relaxes this requirement by employing a finite core size for each vortex on a surface which is influencing a control point in another aurface (unless the two surfaces share the same COMPONENT declaration). This feature can be disabled by setting the core size to zero in the OPER sub-menu, Option sub-sub-menu, command C. This reverts AVL to the standard VL method.

2. Spanwise vortex spacings should be "smooth", with no sudden changes in spanwise strip width. Adjust Nspan and Sspace parameters to get a smooth distribution. Spacing should be bunched at dihedral and chord breaks, control surface ends, and especially at wing tips. If a single spanwise spacing distribution is specified for a surface with multiple sections, then AVL will fudge the spanwise distribution as needed to ensure that a point falls exactly on the section location. Increase the number of spanwise points if the spanwise spacing looks ragged because of this fudging.

3. If a surface has a control surface on it, an adequate number of chordwise vortices Nchord should be used to resolve the discontinuity in the camberline angle at the hingeline. It is possible to define the control surface as a separate SURFACE entity. Cosine chordwise spacings then produce bunched points exactly at the hinge line, giving the best accuracy. The two surfaces must be given the same COMPONENT and the same spanwise point spacing for this to work properly. Such extreme measures are rarely necessary in practice, however. Using a single surface with extra chordwise spacing is usually sufficient.

4. When attempting to increase accuracy by using more vortices, it is in general necessary to refine the vortex spacings in both the spanwise AND in the chordwise direction. Refining only along one direction may not converge to the correct result,

especially locally wherever the bound vortex line makes a sudden bend, such as a dihedral break, or at the center of a swept wing. In some special configurations, such as an unswept planar wing, the chordwise spacing may not need to be refined at all to get good accuracy, but for most cases the chordwise spacing will be significant.

3.2 Mass Input File – xxx.mass

This optional file describes the mass and inertia properties of the configuration. It also defines units to be used for run case setup. These units may want to be different than those used to define the geometry. Sample input xxx.mass files are in the runs/ subdirectory.

3.2.1 Coordinate system

The geometry axes used in the xxx.mass file are exactly the same as those used in the xxx.avl file.

3.3 Run-Case Save File – xxx.run

This file is generated by AVL itself. It can be edited with a text editor, although this is not really necessary. The parameter values in the file can be changed using AVL's menus, and the file can then be written again. Manipulating and using the contents of the run file will be described later.

Capitolo 4

Program Execution

AVL is executed with the "xxx" descriptor as an argument:

```
avl xxx
```

If the three filenames do not obey the recommended xxx.avl xxx.run xxx.mass syntax, the full filenames can be given explicitly:

```
avl avl_file run_file mass_file
```

As the data files are read and processed, a considerable data dump is displayed. If any file has a bad format, the offending data line is displayed, and AVL will stop if the error is fatal.

After the files are processed, the user is put into the main AVL menu: