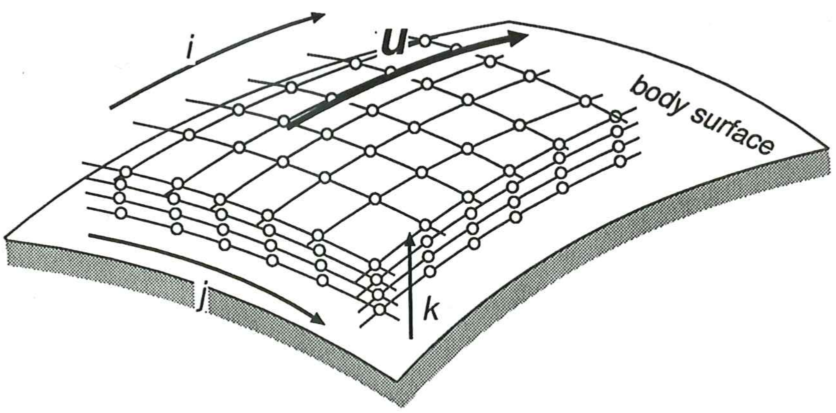
BLAY3D: 3D Boundary Layer Code
User's Guide

M. Storti



Publication CIMNE Nº 158, February 1999

BLAY3D: 3D Boundary Layer Code User's Guide

M. Storti

Publication CIMNE Nº 158, February 1999

International Center for Numerical Methods in Engineering
Gran Capitán s/n, 08034 Barcelona, Spain

BLAY3D: 3D Boundary Layer Code. User's Guide

Mario Storti
Centro Internacional de Métodos Computacionales en Ingeniería (CIMEC)
Santa Fe, Argentina
http://venus.arcride.edu.ar/gtm-eng.html
and
Centro Internacional de Métodos Numéricos en Ingeniería (CIMNE)
Barcelona, Spain
http://www.cimne.upc.es/

January 31, 1999

$Id: b3dman.tex,v 1.15 1999/01/05 14:33:45 mstorti Exp $

Abstract

This is the user's guide for the BLAY3D code. (version 2.2 of December 28, 1998). BLAY3D is a fluid mechanics code for solving the incompressible Boundary Layer Equations (BLE), based on a spectral approximation in the transversal direction and a mesh-less approximation on the surface of the body. It contains a basic description of what are the BLE, the governing equations involved, how they are solved, which is the data and how is it entered in order to be processed by BLAY3D, what kind of result BLAY3D generates, and finally several examples are shown.

Contents

1 The boundary layer approximation

When a body is moving immersed in a fluid, it exerts pressure and viscous forces on it. The whole fluid velocity pattern and forces can be computed through the numerical solution of the Navier-Stokes equations by finite-element or finite-difference discretization methods. However, the computational effort required for these methods are so important that spoils its use for a large class of engineering problems.

When the velocity of the body is high, as measured by the non-dimensional *Reynolds number*, $\mathrm{Re} = UL/\nu \gg 1$ velocity field far from the body approaches the *inviscid velocity pattern*, which can be obtained by the solution of the inviscid equations, namely Euler or potential equations. The computational effort required for the solution of the inviscid equations are much lower than for the Navier-Stokes equations and give very approximated distribution of pressures. However, it is obvious that it doesn't give any approximation to the shear stress, and then to the viscous drag on the body.

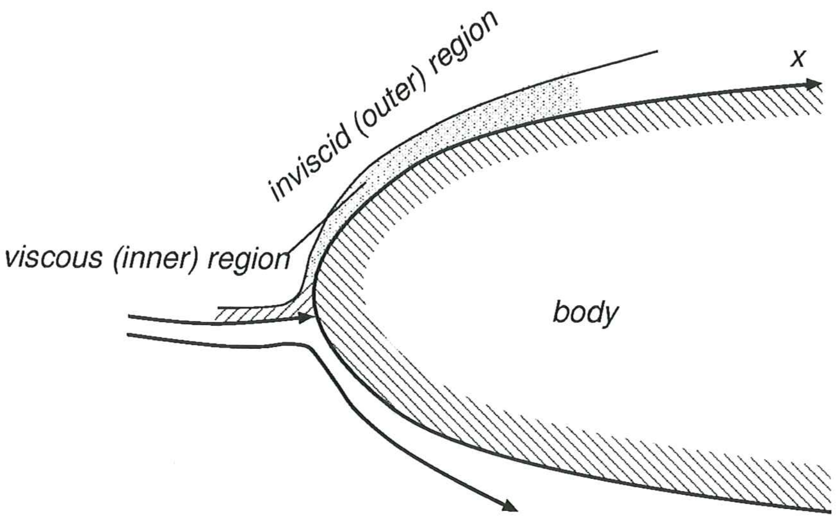


Figure 1: Decomposition of the flow domain in outer nd inner regions.

The *boundary layer theory* [11, 8, 9, 7] asserts that when the Reynolds number is high, the viscous effects are concentrated on a very thin layer of fluid near the skin of the body (see figure 1). Then the full Navier-Stokes problem is decomposed in two simpler problems. First the inviscid (also called "outer") region which can be solved with potential or Euler formulations, and the viscous (also called "inner") region where the "boundary layer equations" are solved. The BLE are very similar to the Navier-Stokes equations but viscous effects in the streamline direction are discarded (this is termed also "parabolization" of the Navier-Stokes equations). This allows the solution of the problem following the streamlines as if the stream-line coordinate were time-like and this results in a large reduction of computing time and core memory requirement. Boundary layer calculations may involve millions of points in the boundary layer at a fraction of the computational resources that would be needed for a similar calculation with the full Navier-Stokes system.

BLE can include turbulence effects. In fact many turbulence models, like the *Prandtl mixing-length* or *eddy-viscosity* model have been developed for the BLE, and then extended to the NSE, so that their application is better justified for the BLE than for the NSE. In addition, some quantities like that are involved in these turbulence models, like the distance to the wall, for instance, are naturally computed in the BLE, whereas they are loosely defined for the NSE.

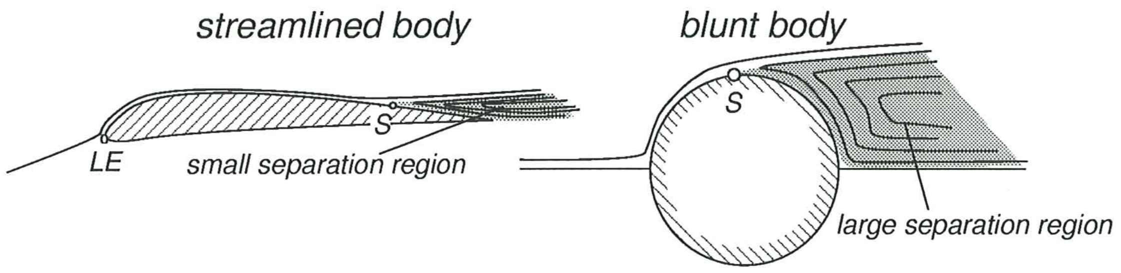


Figure 2: Separation regions for streamlined and blunted bodies.

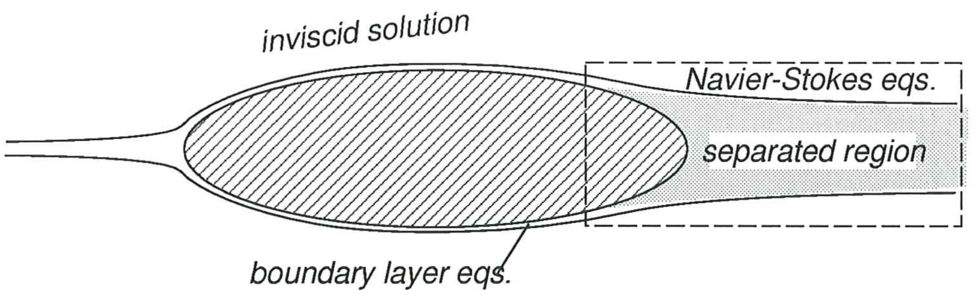


Figure 3: Coupling inviscid flow/boundary layer solution with full resolution of NS eqs..

The boundary layer theory breaks-down if the flow *separates*. However, for rather streamlined bodies the separation region is small, and the skin friction is very low in the separated region, so that computing viscous forces from the BLE is a reasonable assumption. For rather blunted bodies (like a sphere or a cylinder) the separated region is large and significantly perturbs the inviscid flow. In this case the BLE is not directly applicable but it can still be applied coupling all the three, inviscid flow, the BLE and the solution of the NSE in a small region near the rear part of the body.

2 Governing equations

2.1 The 2D case

Let's start with the 2D case. The BLE are solved in a curved strip attached to the body (see figure 4). Taking a curvilinear coordinate system with x as the curvilinear coordinate and y as the normal coordinate, the strip is bounded by $y = 0$ (the body skin) and extends in the normal direction to $y = y_{\max}(x)$ where $y_{\max}(x)$ is computed automatically by the code and is related to the local width of the BL. In the longitudinal direction, it starts at the stagnation point S ($x = 0$) and proceeds in the sense of the fluid. If the body is symmetric about its chord (like a symmetric airfoil) and the unperturbed flow is parallel to this axis (null angle of attack) then we have to solve the BLE only on one side of the body. In any other case, we have to solve the BLE on each side (they are independent problems) and then sum up the viscous forces from each side.

The 2D BLE are

$$uu_{,x} + vu_{,y} = \nu u_{,yy} + U_{\text{ext}}\, U_{\text{ext},x} \tag{1}$$

$$u_{,x} + v_{,y} = 0 \tag{2}$$

where u, v are the streamwise and normal components of velocity, ν is the kinematic viscosity and U_{ext} is the *outer velocity*, i.e. the velocity immediately outside the boundary layer that is computed from the inviscid flow. The computation starts at the stagnation point and are continued in the direction of the flow until either the streamlines exit the surface or the flow separates. To start computations, an analytic solution based on similarity assumptions is assumed near the stagnation point and matched to the numerical solution.

2.2 The 3D case

The 3D boundary layer eqs. are considerably more complex than the 2D case [15]. In the 2D case, the curvature of the surface does not enter (unless indirectly via the pressure distribution) whereas in the 3D equations they do appear through the *metric tensor* and other tensorial characteristics of the surface. The 3D equations are properly described using tensorial notation [11]. *You don't need to understand or either read the following material. It is included only to the sake of completeness.*

x^1, x^2 denote a system of curvilinear coordinates (non necessarily orthogonal) on the surface of the body, η denotes the normal coordinate, δ the local thickness of the boundary layer, α, β, γ are indices running over the surface coordinates, X^α, X_α denote the contravariant and covariant components of a given vector X (they are equivalent for orthogonal systems), $g^{\beta\gamma}$ is the contravariant metric tensor, u^α are the contravariant components of velocity and v the normal component of velocity. $X_{,\alpha}$ denotes the *covariant derivative* of X, an extension of the standard partial derivative to curvilinear systems of coordinates. The 3D boundary layer equations are,

$$u^\alpha u^\beta_{,\alpha} + v\frac{\partial u^\beta}{\partial \eta} = -\frac{\nu}{\delta^2}\frac{\partial^2 u^\beta}{\partial \eta^2} + g^{\beta\gamma}\, p_{,\gamma} \tag{3}$$

$$\frac{1}{\delta}\,(\delta u^\alpha)_{,\alpha} + \frac{\partial v}{\partial \eta} = 0 \tag{4}$$

3 Input data to BLAY3D

Basically, the user has to give to BLAY3D

- Physical parameters, like viscosity, density, ...

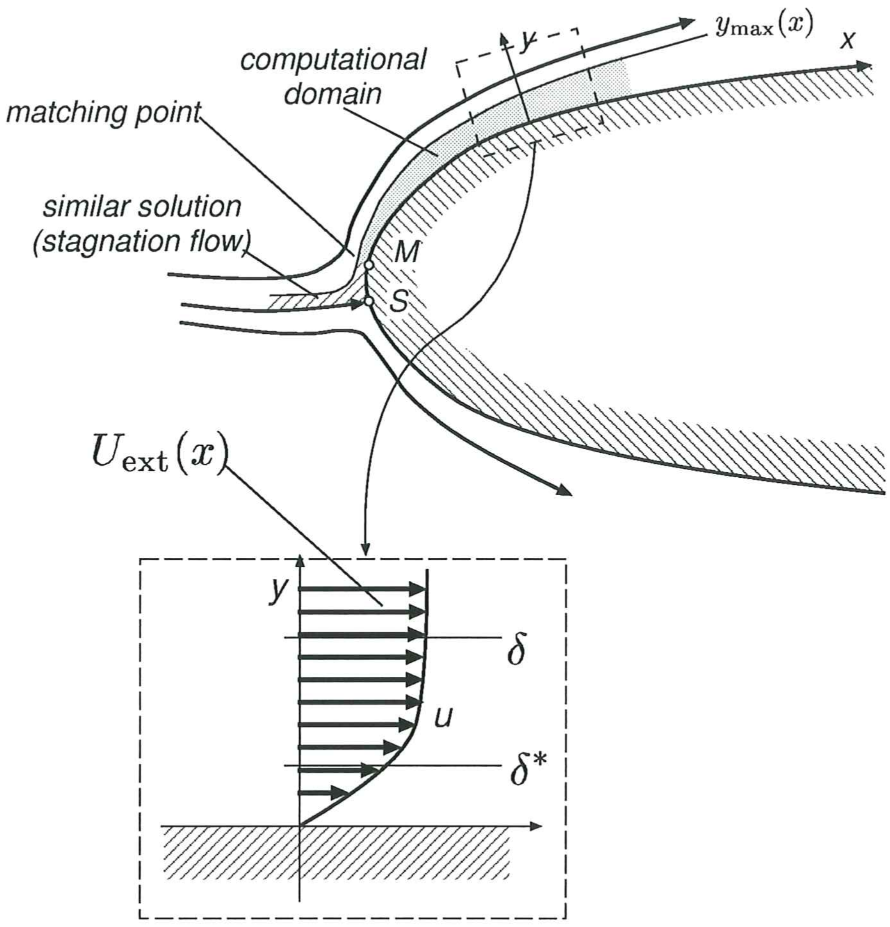


Figure 4: Coordinate system and geometrical setup for the description of the solution of the BLE.

- Parameters controlling the numerical modelization, like the number of streamlines, number of parameters representing the velocity profile at a given surface point, ...
- Discretization of the surface of the body, given by a set of nodes on the surface and a triangular panelization.
- The starting points of the streamlines.

We describe here the syntax and organization of the files that the user should prepare, while in §4 we describe in depth the numerical method and the meaning of each of the variables that entered by the user.

This data is stored in four files. A master file (as a convention, normally with extension *".cdl"*) contains all the parameters, whereas the other three contains, respectively, the nodes coordinates, panel connectivity, and velocities at the nodes. The name of these three files is free and is passed to the program through three string variables in the master file.

The master file is format free. It is written in a language called CDL, that is easily interchangeable with a format called NetCDF. NetCDF is a standard for storing data and results used in the context of large scientific applications. However, the user don't has to learn anything about CDL or NetCDF, the CDL format is rather self-explaining. For instance, this is a typical CDL file

```
netcdf blay3d {

  // $Id: b3dman.tex,v 1.15 1999/01/05 14:33:45 mstorti Exp $

dimensions:
  nz=31,nd=3,maxchar=20;

variables:
  int icase,
    na, m_expo, nx, initial_axf, nprint, nitmax, interp,
    auto_stag;
  double r0, ddfi, anuvisc, xstag(nz,nd), e1(nz,nd),
    max_normal_dist,auto_stag_relax,smoothing_parameter,
    xcenter(nd),density;
  char nodes(maxchar),elems(maxchar),vels(maxchar);

data:
  nodes="wigley.xyz";
  elems="wigley.top";
  vels= "wigley.vel";
  auto_stag=0;
  icase=0;
  na=6; m_expo=5; nx=5000; initial_axf=0; nprint=10; nitmax=20;
  interp=1;
  r0=0.01; ddfi=0.01; anuvisc=1.0 ;
  auto_stag_relax=1.0;
  smoothing_parameter=1.0;
  xcenter=0.,0.,0.;
  density=1;
  xstag= // start. of streaml. (3 reals per streamline)
    -8.00,  0.00, 0.00,
    -8.00,  0.00, -1.00;

  e1=   // initial direction (3 doubles per streamline)
   1.00000,  0.00000, 0.00000,
   1.00000,  0.00000, 0.00000;
}
```

All the data is enclosed in a `netcdf blay3d { ... }` environment. Files preceded with *double-slash* ("`//`") are discarded, i.e. they are comments (like in C++). There are three sections of the file, identified by (in this order) labels `dimensions:`, `variables:`, and `data:`.

In the `dimensions:` section, the dimensions of arrays used to enter the data are defined. For BLAY3D the user has only to define `nz`, the number of streamlines, `nd`, the number of dimensions (this is always `nd=3`), and `maxchar`, the maximum number of characters the string containing the filenames can have (we use `maxchar=20`).

Other parameters are defined through variables. This variables have to be declared (with type) previously in the `variables` section, and then assigned a value in the `data` section. Variables can be `int`, `double` or `char`. So far, the only `char` variables are `nodes`, `elems` and `vels`, the names of the files containing the respective arrays defining the geometry and inviscid velocity field.

The rest of the quantities describing the data are:

- `double anuvisc` (default=1) Kinematical viscosity (ν) of the fluid.
- `int auto_stag` If equal to 1, BLAY3D tries to detect automatically the position of the stagnation point or stagnation line. The position given by the vector `xstag` is taken as an initialization of the search process.
- `int auto_stag_relax` (default=1) Tries to stabilize the process of automatic detection of the stagnation point. It must satisfy 0 ≤`auto_stag_relax`≤ 1. The value `auto_stag_relax`=1, means no relaxation. The smaller this parameter is, more stable is the convergence process (but also slower).
- `int auto_stag_max_iter` (default=20) Maximum number of iterations allowed for the computation of the stagnation point or stagnation line.
- `double ddfi` (default=0.01) This parameter controls the spacing between points on the streamline. The smaller it is, the closer are the points on the streamlines.
- `double density` (default=1) This is the density of the fluid.
- `char elems(maxchar)` (no default value) String defining the file containing the topology (also called "connectivity") of the panel mesh describing the surface. This file should contain one line per panel with the three nodes defining the panel.
- `double e1(nz,nd)` (no default value) This array defines the directions of the starting streamlines. §4.3

- `int icase` (default=0) This defines the case, i.e. the geometry and inviscid velocity field, to be studied. If set to 0, it means that both are given by the user in the discrete form described above. `icase`> 0 selects on of several internal defined cases described in section 4.6.

- `int initial_axf` (default=0) If $= 1$, means that the calculation starts at a stagnation point, if $= 0$ the calculation starts at a stagnation line (see section §4.3).

- `int interp` (default=0) If$= 1$ determines that the user enters only the first ($k = 1$) and last ($k =$`nz`) of the stagnation points and directions and the code interpolates the intermediate values.

- `int m_expo` (default=5) Defines the number of points ($N_\eta = 2^{\texttt{m\_expo}}$) of integration of the spectral method for the normal direction. Normally N_η should be several times greater than the number of parameters `na` defining the boundary layer. For instance, `m_expo`$= 5$ ($N_\eta = 32$) and `na`=6 is a common (low precision) combination. A high precision one is `m_expo`$= 7$ ($N_\eta = 128$) and `na`=12.

- `double max_in_plane_distance` (default=0.5) This sets the threshold for marking streamlines as separated when a streamline goes beyond the last panel. The value is relative to the local panel size. (see §4.7)

- `double max_normal_dist` (default=0.5) Maximum (relative to local mesh size) normal distance allowed when tracing streamlines. (§4.8)

- `int na` (default=6) The number of terms describing the each component of the velocity in the boundary layer profile. The higher this number is, the calculation is more precise but the CPU time required is higher.

- `int nitmax` (default=20) The maximum number of iterations allowed to solve to the prescribed tolerance (10^{-10}) the non-linear equation at each point. If the convergence of the Newton iteration is poor, and the prescribed tolerance is not reached within `nitmax` iterations, then a warning is issued and the computation proceeds.

- `char nodes(maxchar)` (no default value) String defining the file containing node coordinates. This file should contain one line per node with the three coordinates of the node.

- `int nprint` (default=10) Results are printed in the Fortran unit 10 for each streamline only each `nprint` longitudinal stages in order to avoid too large amounts of output.

- `int nx` (default=5000). Number of longitudinal stages to be computed at maximum. It may happen that the boundary layer separates before.
- `int only_streamlines` (default=0). If $= 1$ the streamlines are calculated, but the BLE are not solved. Useful to make a first run to in order to check if the external velocity field is properly defined.
- `double r0` (default=0.01) Distance from the stagnation point (or line) to the first points in the streamline.
- `double smoothing_parameter` (default=1). (See section §4.2) This parameter controls the amount of artificial diffusion added in order to smooth data and prevent premature separation of the boundary layer due to oscillations or lack of regularity in the inviscid velocity field. Setting it to zero means no artificial diffusion added, and setting to one means that the maximum amount of artificial diffusion is to be added.
- `double xcenter(3)` (default=0.,0) Coordinates of the point with respect to which the components of torque are computed.
- `double xstag(nz,nd)` (no default value). Coordinates of the stagnation point or a series of points on the stagnation line.
- `int warn_on_not_unique_nearest` (default 0). If equal to 1 give a warning message each time that there are more than one panel which is the nearest to a given point. (see §4.8)

4 The numerical method

The BLE are solved mainly by integral methods or the FDM (Finite Difference Method) [10, 4, 15]. Integral methods are based on a very simple representation of the velocity profile at a given point of the surface using two parameters, namely the boundary layer thickness δ and a shape parameter. Most of them are reminiscent of the original von Kàrman and Pohlhausen method, and are simple and fast, but not convergent, i.e. there is not a parameter of refinement option that can improve precision while increasing the computational effort (like the number of discretization points in the FDM).

The method of discretization used in BLAY3D is a *spectral discretization* in the normal direction, combined with a *mesh less* discretization in the surface coordinates. As both methods can be interpreted as derivations of the FDM and, presumably, users are more likely familiar with the FDM than with the spectral or mesh-less discretizations, we will explain briefly the a typical FDM and then pass to discuss some details of them.

Solution of the BLE by the FDM is rather straightforward and follows very closely the technique used to discretize the Navier-Stokes equations. A 3D grid whose typical node is indexed as (i, j, k) is used (see figure 5) and assuming that the i index is streamline-oriented, the field quantities at the $i + 1$ plane of nodes are computed from the previous

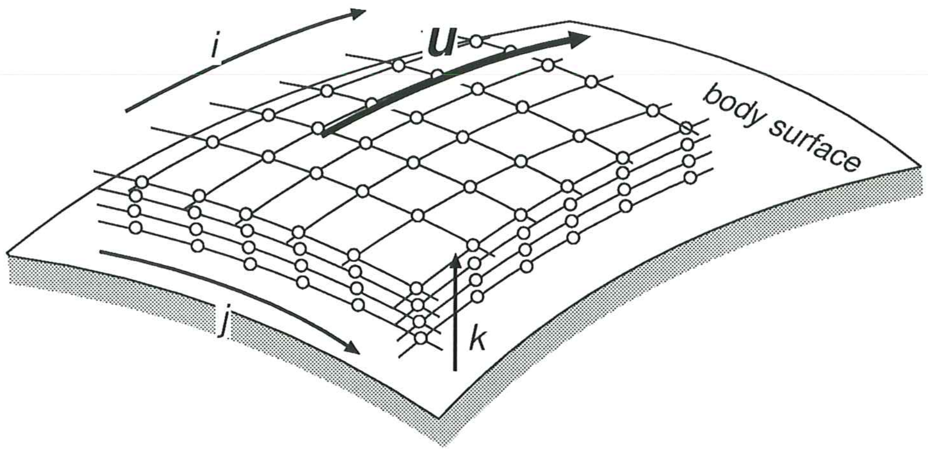


Figure 5: Three dimensional grid for FDM computation

ones using standard FDM approximations as if the i direction were time-like. Off course, the velocity doesn't need to be perfectly orthogonal to the $i=\text{cnst}$ plane, in fact, in 3D the velocity direction may very along a vertical column of nodes $i,j=\text{cnst}$.

FDM is based on replacing partial derivatives by finite difference approximations of the form

$$\left(\frac{\mathrm{d}f}{\mathrm{d}x}\right)_j \approx \frac{f_{j+1}-f_{j-1}}{2h} \tag{5}$$

where f_j denotes the nodal value of the numerical approximation at the j node. Replacing this kind of expressions in the governing equations one obtains a system of equations that determines the numerical solution. Eq. (5) is a *low order approximation*. *Higher order approximations* can be developed which will give more accurate results for a given fixed computational effort, provided that the solution of the continuum equations is regular enough. It can be shown that the solution to the BLE are highly regular, so that it's appealing the use of higher order approximations. In BLAY3D a *spectral method* is used for the discretization in the normal direction. Spectral methods [1, 3, 14, 2, 13] can be thought as finite difference approximations of *infinite order* and can give very accurate results with a very low number of points in the discretization.

On the other hand, the FDM method requires that the computational grid be *square structured* or either *smooth curvilinear structured* (see figure 6). This last one means that it can be smoothly mapped to a square constant step grid. In BLAY3D the mesh is not generated by the user but it results from integration of the inviscid streamlines. The points on the surface have, then, a topological structure where we can assign to them two indices, the *streamline index* and the position of the point in the streamline (from the stagnation point to the rear of the body). This results in a *topologically structured* mesh, that means meshes that have a (i,j,k) structure, but can't be mapped smoothly onto a square mesh. They look as if a random shift will be applied to the nodes on a curvilinear smooth mesh. The *mesh-less method* is an extension of the FDM that allows such kind of meshes. [5, 6]

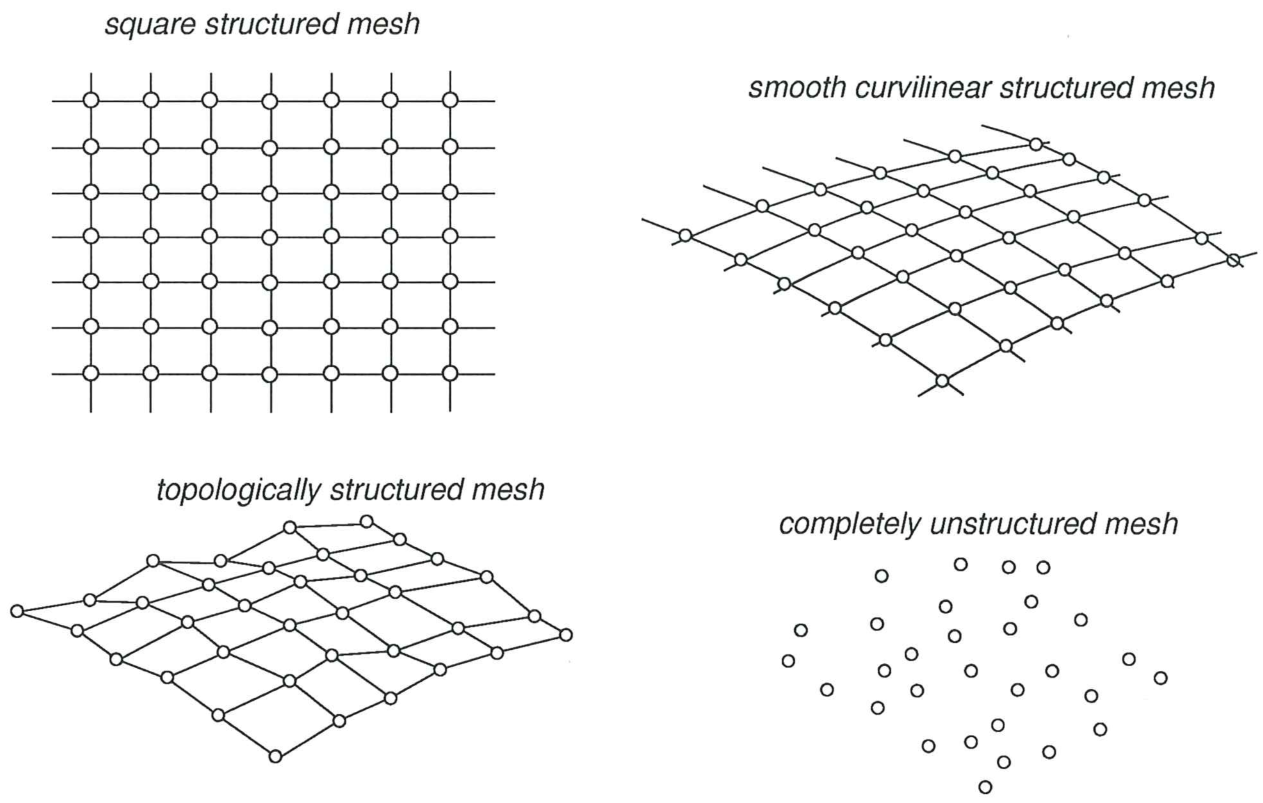

square structured mesh
smooth curvilinear structured mesh
topologically structured mesh
completely unstructured mesh

Figure 6: Structured meshes, both square and curvilinear.

4.1 Critical time step

Consider the advection equation

$$\frac{\partial \phi}{\partial t} - U\frac{\partial \phi}{\partial x} = 0 \tag{6}$$

The solution of this equation is of the form $\phi(x,t) = f(x - Ut)$, i.e. the values of ϕ are constant on the characteristic lines $x = x_0 + Ut$. Now consider its solution by a numerical method on a constant step grid $\Delta x, \Delta t$ with explicit time integration. Assume that the initial condition is

$$\phi(x, t=0) = \begin{cases} 1; & x \leq 0; \\ 0; & x > 0; \end{cases} \tag{7}$$

and the scheme is advanced several, let's say N, time steps, so that the continuum solution at time $t_N = N\Delta t$ is

$$\phi(x, t=N\Delta t) = \begin{cases} 1; & x \leq UN\Delta t; \\ 0; & x > UN\Delta t; \end{cases} \tag{8}$$

But, as the scheme is explicit, it can't advance further than one element per time step, so that the position of the advancing front $UN\Delta t$ must be at the left of Nh, i.e. the follow restriction must apply

$$UN\Delta t \leq Nh \tag{9}$$

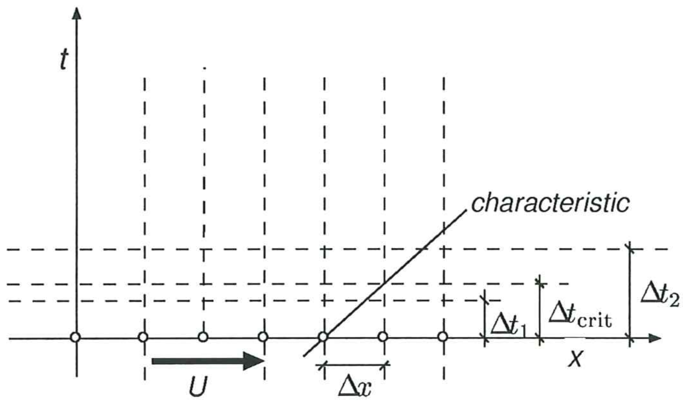


Figure 7: Critical time step in solving the advection equation.

so that

$$\Delta t \leq \Delta t_{\text{crit}} = h/U \tag{10}$$

This condition is often stated as CFL $= \Delta t U/h < 1$ where CFL is the Courant-Friedrichs-Lewy non-dimensional number representing the ratio between the velocity of the fluid and highest velocity attainable by the scheme $h/\Delta t$. Using the scheme with time steps that exceed the critical one leads to instabilities that growth exponentially in time and spoil the solution.

A similar restriction applies to the BLE. If we consider the 3D BLE and consider a mesh on the surface with (almost) constant $\Delta x, \Delta z$ (see figure 8), then the advective term is of the form

$$u\frac{\partial \phi}{\partial x} + w\frac{\partial \phi}{\partial z} = \text{r.h.s.} \tag{11}$$

where ϕ is any of the transported quantities. If we consider x as a time-like coordinate, then the CFL restriction reads now

$$\Delta x < \left|\frac{w}{u}\right| \Delta z \tag{12}$$

Now for a certain position x, z on the advancing front, the values of w, u vary on the vertical position y, and are not known a priori, so that finding the critical advancing step Δx_{crit} is more complex than for the advection-diffusion equation. In addition, in BLAY3D the user controls the quantity **`ddfi`** which represents the difference in potential between consecutive layers, but only controls indirectly Δx. However, the following facts can help at the moment of choosing **`ddfi`**:

- As BLAY3D tends to use a mesh with layers perpendicular to the inviscid velocity vector, the restriction comes due to variations of the velocity vector along the normal velocity. In particular, near a separation point (like S in figure 16) the situation is critical, since the velocity vector varies 90° along the normal direction.

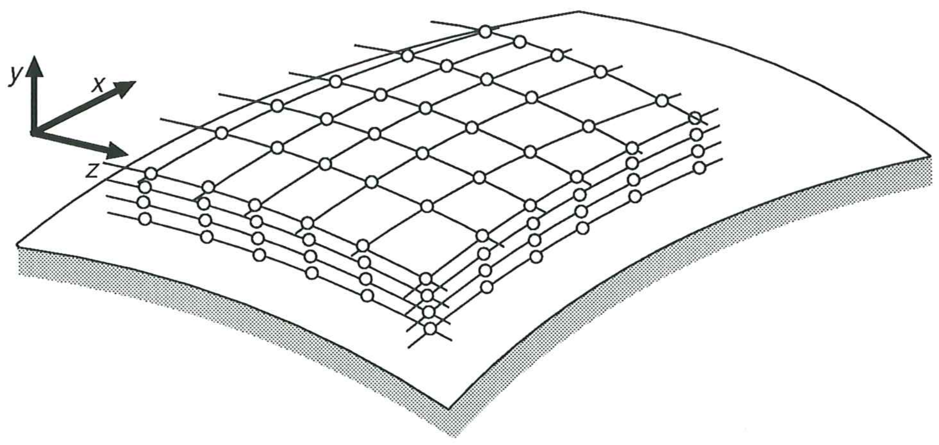


Figure 8: CFL restriction for the BLE

- If you increase the number of streamlines (**nz**), then the spacing Δz between them decreases and the critical time step also decreases.
- If you observe that the solutions tends to become noisy, try to lower `ddfi`. If this effect tends to disappear, then this is very likely a stability problem.

4.2 Effects of the discrete representation of surfaces

Even if BLAY3D allows the computation of some reduced set of analytical cases, mainly for diagnostic and experimentation, the practical use of it is for the computation of boundary layers for complex geometries defined by discrete representation with panels and with external velocities coming from potential or Euler calculations via numerical methods like FDM, FEM or panel methods.

Such discrete representations introduces noise in the variables. Consider for instance data generated by a potential calculation in figure 9. Depending of the potential model used, we can obtain a discontinuous discrete velocity field constant per panel or linear constant velocities at each panel or a continuous velocity field linear per panel. The first case can be converted to the second case via standard projection methods. However, in all cases the numerical representation of the velocity field is far less regular (this means less smooth) than the real one. In the worst case, if the numerical inviscid model is not correctly adjusted, or the discrete representation of the surface is poor, oscillations in the numerical velocity field can exist. This lack of regularity can promote premature separation of the boundary layer.

In general, those solver of the BLE that are less accurate tend to separate lately, and even to not to separate in some situations where the real flow does separate. In

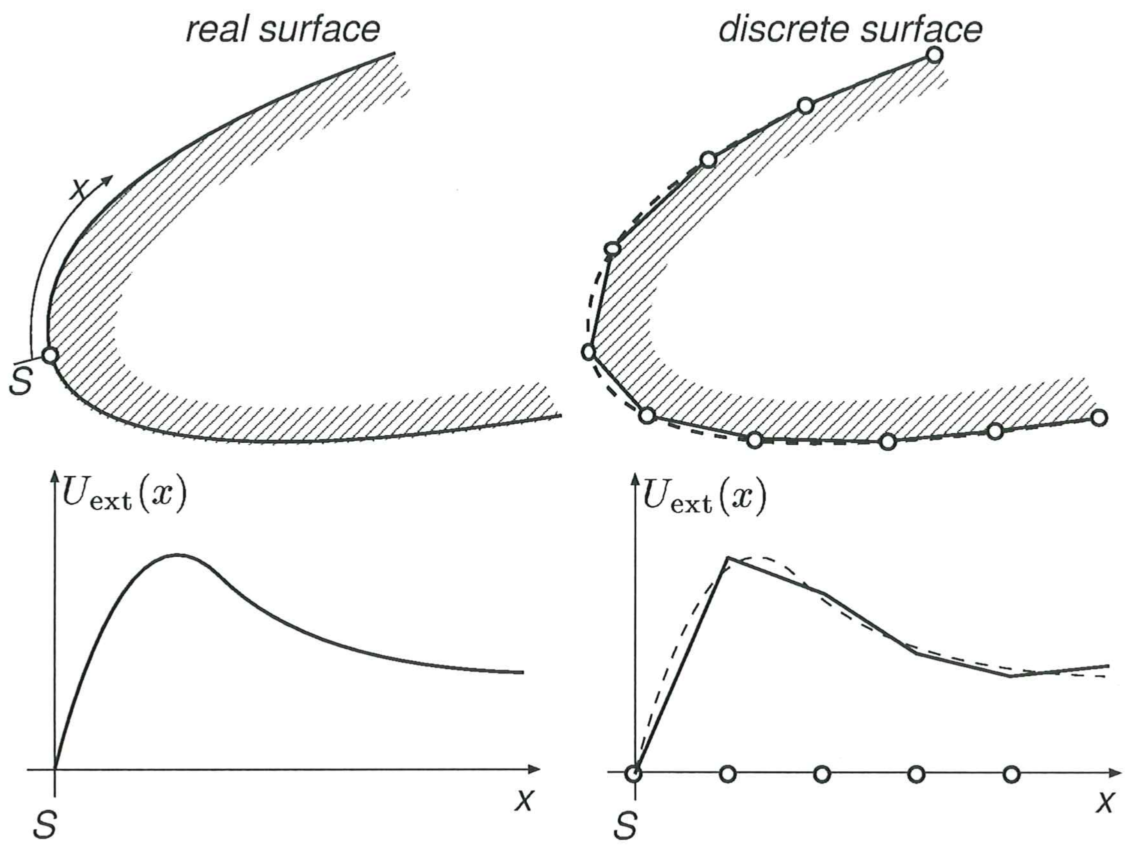


Figure 9: Discrete representation of surfaces and external velocity (inviscid) fields

brief, less accurate BLE solvers tend to be more stable with respect to separation. The spectral/mesh-less combination of BLAY3D is a very accurate solver, and then it is very sensitive to this lack of regularity in the inviscid velocity field. In order to prevent premature separation, an artificial diffusion mechanism that tends to smooth the results has been added. This mechanism is controlled by a real parameter called `smoothing_parameter`. Setting it to zero means no artificial diffusion added, and setting to one means that the maximum amount of artificial diffusion is to be added. We recommend setting it to 1 in all cases where discrete data is involved.

The way BLAY3D determines to which panel a given point in the streamline belongs is discussed in §4.8.

4.3 Initialization of the solution process. Stagnation points and lines.

As described before, the solution of the BLE are made in a time-like manner starting from the stagnation point and following the streamlines to the rear of the body. Roughly

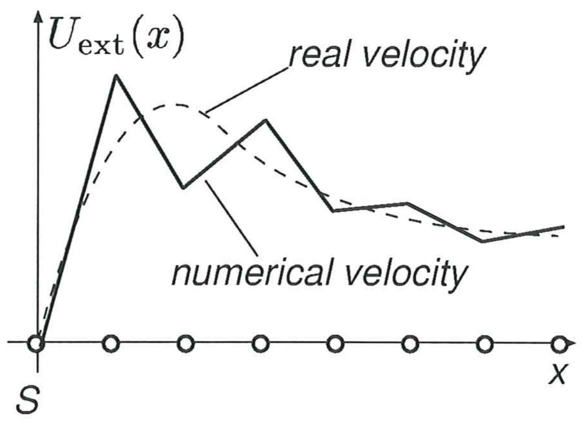


Figure 10: Oscillations in the numerical inviscid velocity field.

speaking, the stagnation point is the point where an inviscid streamline impinges on the front of the body. This implies that all three components of the velocity are null at the stagnation point. We call this a stagnation point and is more likely to occur in the front point of a blunt body as depicted in figure 11. Near the stagnation point, the iso-potential curves can be approximated by ellipses. If the body is axially symmetric, then the iso-potentials are circles. On the other hand, for bodies that span in the transversal direction, like wings, it is more likely to have iso-potentials that are very elongated ellipses. In the limit, for cylindrical bodies, i.e. bodies of constant section spanning in the transversal direction, the ellipses degenerate in two lines and the velocity is null in a whole *"stagnation line"* (see figure). In BLAY3D the variable `initial_axf` selects between the limit cases

- `initial_axf=0` selects stagnation lines.
- `initial_axf=1` selects stagnation point.

The BLE are a set of ODE and they have to be initialized. However, the BLE are a particular case, since in general they are automatically started by matching the solution with a solution obtained by similarity assumptions in a small region near the stagnation point or line. The user must give the radius `r0` of the region where the similarity solution is assumed.

In addition to this, the user must enter the initial point x_k and direction $(\hat{\mathbf{e}}_1)_k$ of each of the streamlines to be computed. (variables `xstag` and `e1`). (The denomination $\hat{\mathbf{e}}_1$ comes from the fact that we take this vector locally as the axis of the first coordinate in a local coordinate system). If the set of initial points lie on a straight line, then the user may enter only the first and last of them and BLAY3D interpolates for the rest, this is controlled with the logical flag `interp`.

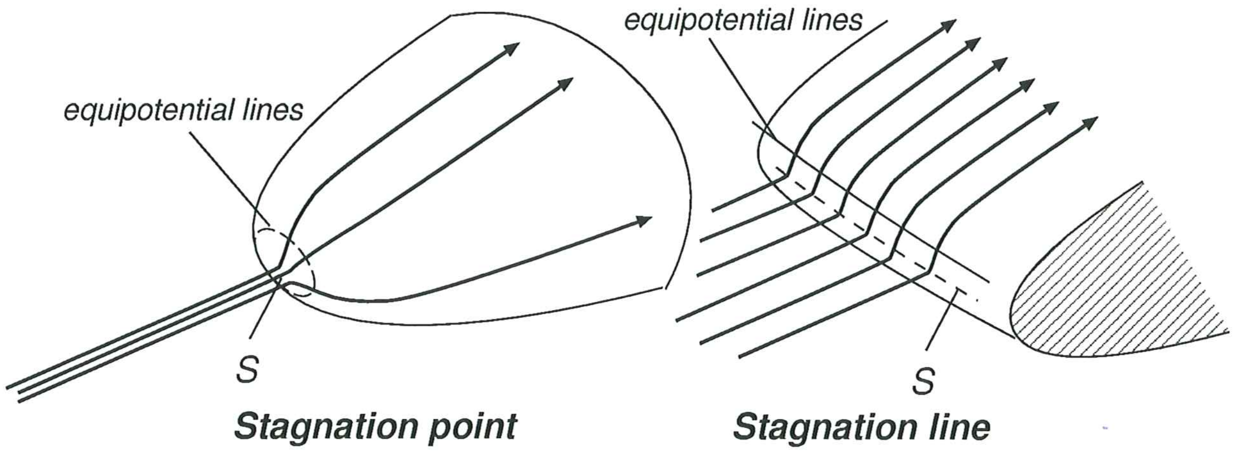


Figure 11: Different types of stagnation.

4.4 Automatic detection of stagnation points and lines.

For blunted bodies, the location of the stagnation point depends on the angle of incidence of the external velocity field. For instance, for a symmetric hydrodynamic profile aligned with the flow (see figure 13) the stagnation point is on the symmetry plane, but as the angle of incidence of the flow increases the stagnation point moves towards the intrados.

If `auto_stag` is set to 1, then the position of the stagnation point and initial direction of the streamlines may be only approximate, and BLAY3D corrects them from the velocity field entered. This process is based on a type of secant iteration and may fail when the entered point is too far from the actual position or if the velocity field is too noisy near the stagnation point. This process is controlled by the variables `auto_stag_relax` and `auto_stag_max_iter`.

4.5 Sharp leading edges.

So far, we described the stagnation point or line as a location of points on a smooth surface where the velocity is null. However, in some idealized cases the stagnation region may be a singular point or line of the surface, as for the leading edge (LE) of a sharp wedge or the vertex of a cone (see figure 14). Provided that the angle β enclosed by the wedge or cone is non-null, the velocity behaves like $U_{\rm ext} = C\,x^m$, with $m > 0$, where x is the distance to the edge. This means that the velocity tends to zero at the LE, and then these are stagnation regions. These cases have also a similarity solution and can also be "automatically started". BLAY3D computes automatically and prints the values of C and m. A particular case is the flat plate, for which $U_{\rm ext}$ =cnst. In this case the LE is not a stagnation point, but the similarity solution is still used. This case is very common in naval hydrodynamics for the case of sharp bows as, for instance, the Wigley hull. (see §6.6)

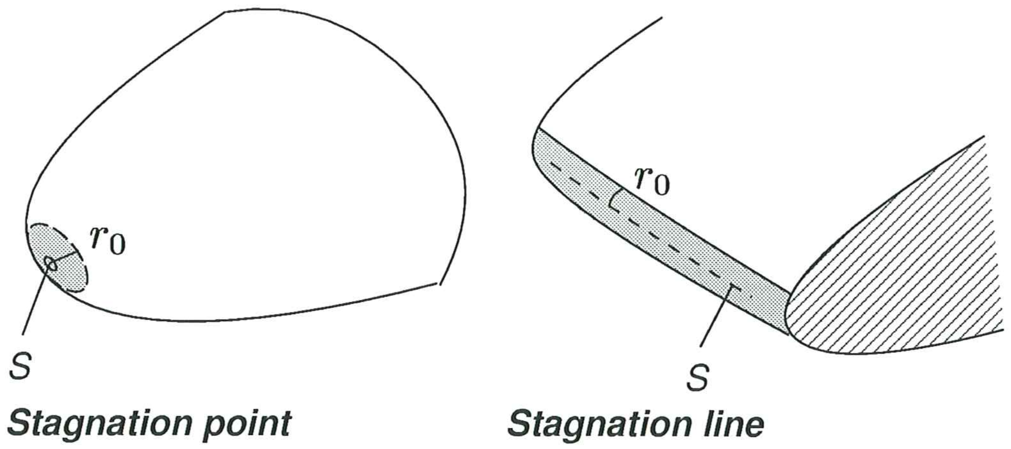


Figure 12: Initial region to be computed with a similarity solution.

4.6 Built-in cases

Several well known cases, like the flat plate, sphere and circular cylinder, are built-in in the Fortran code, so that the user can run them without having to generate the mesh (coordinates and topology) and inviscid velocity field files. This can help in getting some experience with the code, as well as to experience with some advanced features without having to write the files.

The cases included are,

- `icase=2`: Flat plate. Blasius solution. (see §6.1)
- `icase=3`: Sphere (see §6.2)
- `icase=4`: Circular cylinder (see §6.3)
- `icase=5`: Prolate ellipsoid (axis ratios 5:1:1). Inviscid velocity fitted with 3rd order polynomial. (see §6.4).
- `icase=6`: 2D elliptical cylinder. Inviscid velocity fitted with 4rd order polynomial.

They are deeply discussed in the *Numerical Examples* section (§6).

4.7 When computations are stopped?

Calculations are stopped when boundary layer separation is detected, otherwise they continue until the specified number of steps (variables `nx`). Boundary layer separation is detected when the longitudinal component of the skin friction is null. This is true in 2D but not in 3D, but unfortunately there are not other means to detect it. The boundary

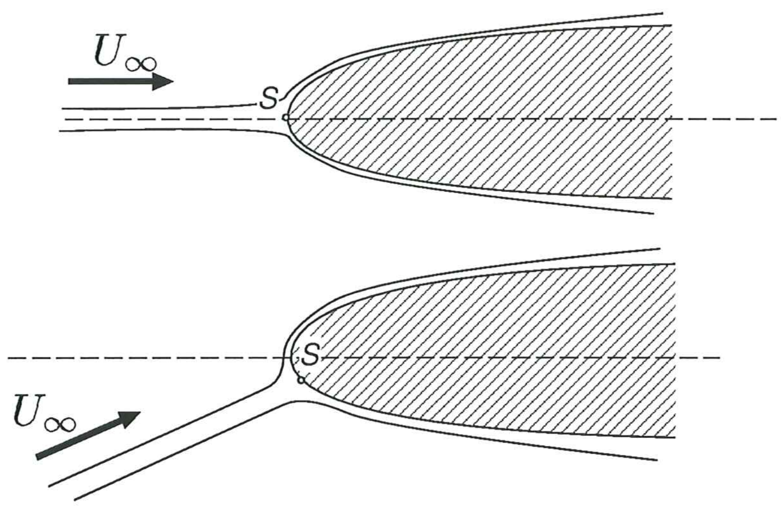


Figure 13: Stagnation points depends on the incidence angle.

layer separates at a curve called the *"separation line"*. Consider a separation bubble as in figure 16. Assume that the flow is symmetric about a plane perpendicular to the surface. Then for an inviscid streamline like "*a*" at the symmetry plane the situation is more like in the 2D case, i.e. the longitudinal skin friction at the separation line at that point is null. However the skin friction on the separation line tends to increase from this *"singular separation point"* and is parallel to the separation line. So that, for an inviscid streamline like "*b*", the skin friction at the separation line is non-null, and the longitudinal component of the skin friction is null at a point which is behind the separation line, i.e. in the separated region.

Stopping calculations the whole calculation when one streamline separates is undesirable, since in a large number of cases the flow can separate locally in a certain portion of the advancing front, whereas the flow remains attached on the rest. Stopping the calculations would be reflected in an underestimation of the drag, since the drag on the attached part behind the advancing front at the separation point is not taken into account. So that it is desirable to allow streamlines separate individually and continue the computation with the others. When the computation of a streamline is continued behind the separation line the system of ODE's becomes unstable and the non-linear system of equations does not converge, so that we detect separation of a streamline also when this occurs. In the ideal case the situation is like in figure 17, where lines are marked as separated as they arrive at the separation line.

There are yet another two cases where we mark streamlines as "separated". If a certain streamline has all of their neighbor streamlines separated, then it can not continue because it needs at least 2 neighbor streamlines in order to compute the transverse derivatives that appear in the system of ODE's (see figure 18). The same happens if it has only one neighbor streamline.

The other case is when the streamline exits the surface. This can happen for very

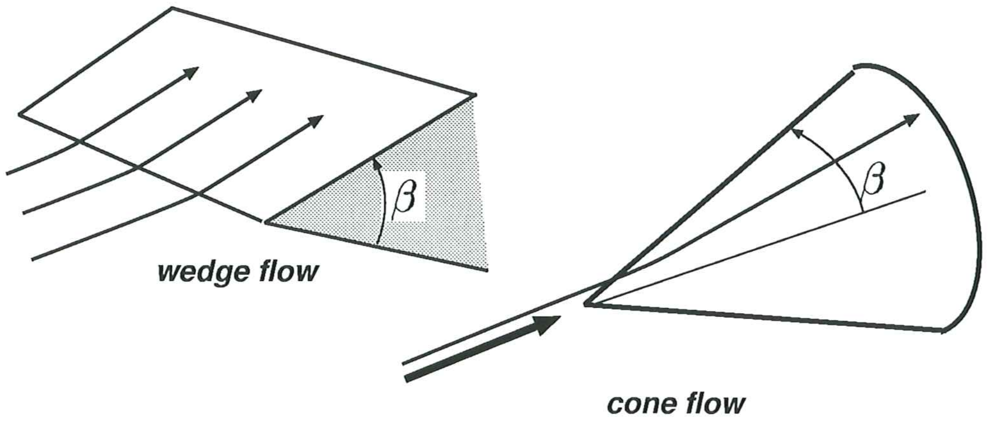


Figure 14: Stagnation region for sharp leading edges and cones.

streamlined bodies when no physical separation is produced. See for instance figure 19, where flow around a very streamlined ship hull is shown. The flow does not separate and then calculations are stopped when BLAY3D detects that streamlines are too far from any panel on the surface. Off course, this is not separation in the true sense, we use here the term separation to indicate that calculations will be stopped for that streamline. This is controlled by a parameter `max_in_plane_distance` which is the maximum distance $d_{\max}$ allowed. If you set it too high, BLAY3D will mark the streamlines as separated too lately, whereas if you set it too low the streamlines can be marked as separated for very small deviations from the surface. This value is relative to the local size of the panels and the default value is 0.5.

4.8 How the panel mesh and inviscid velocities are used.

Each time BLAY3D computes a new point for each streamline, it has to find on which panel it is, and then interpolate the external velocity field to that point. As a rule, BLAY3D chooses the panel which is nearest to the point. An inefficient implementation of this algorithm could results in inadmissible computation times for large meshes, so that, among other things, BLAY3D has a "caching" strategy. For each streamline, it stores ("caches") the panel where the last point in the streamline was. When the new point begins to be processed, BLAY3D first check if the point is on the cached panel. If this is so, a hit is produced, if not the nearest panel is searched in the long manner. If a large percentage of hits are obtained, then the computer time is greatly reduced. BLAY3D prints a resume of the cache statistics at the end of the computation.

Due to numerical errors in the integration of the streamlines, the points can not lie exactly on the panel surface and then there are points which have not a unique nearest element (see figure 20). In this situation BLAY3D selects among the possible panel candidates according to some criterion. Then, if the variable `warn_on_not_unique_nearest` is set to 1 sends a warning message. This can help you if you think that there is something

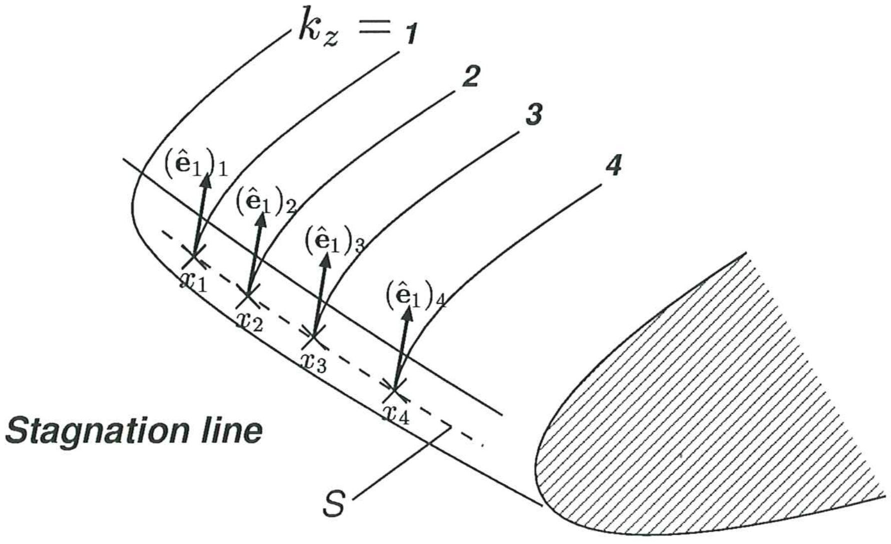


Figure 15: Data for the location and starting direction of each streamline.

wrong with the mesh. This problem is more likely to occur when the angle α between two adjacent panels becomes higher, and then tends to disappear with refinement.

If the distance from a point to the nearest panel is too high, an error condition is issued and the program stops. This may happen if there are errors in the surface or velocity field, or else if the mesh is too coarse or the velocity has too high normal components.

5 Output from BLAY3D

BLAY3D outputs a lot of information to the standard output, including

- Information on current version of BLAY3D
- Interpretation of the variables entered
- Parameters of the similarity expansion at the stagnation region (see §4.5).
- Information on the process of auto-detection of the stagnation region. (see §4.4).
- Information on solution of the non-linear problem in the similarity region.
- Information on solution of the ODE and convergence of the non-linear system of equations at each point.

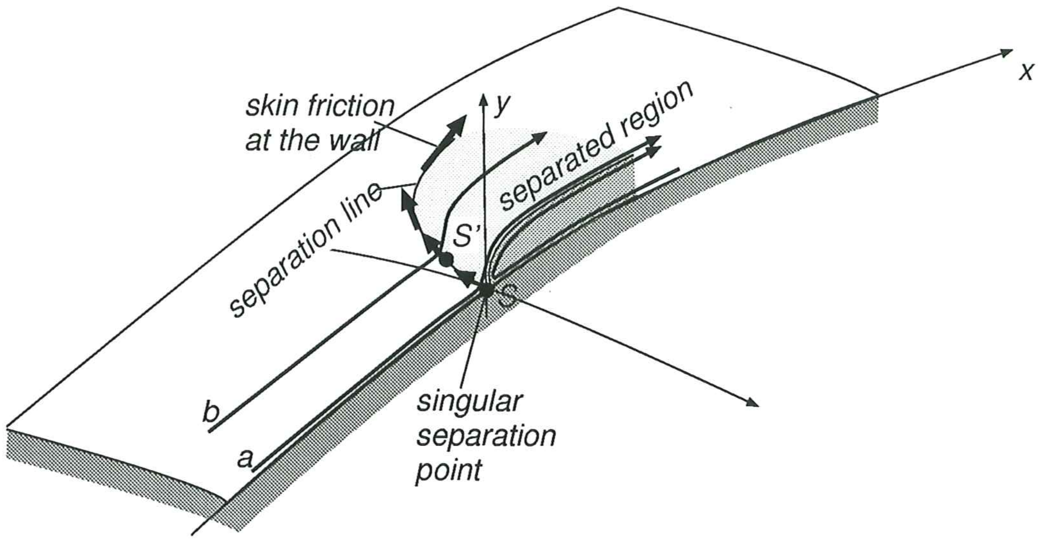


Figure 16: 3D separation bubble.

- Notification of streamline separation.
- Total values computed, including area, force and moment.

In addition, on the Fortran unit 10, one record per point is written containing the following information

- coordinates (3 reals)
- displacement thickness (1 real)
- shear stress (3 reals)

Even if the streamline separated, BLAY3D continues to compute the inviscid streamline position, and results are written to the Fortran unit 10 with displacement thickness and shear stress set to zero.

6 Numerical examples

In this section we present a series of examples using the BLAY3D code, putting the stress in the use of the code, rather than in the validation. Validation of the code is discussed in a separate report [12]. For each example a directory containing the CDL file and the geometry and velocity files needed is provided. Also, the output file (or part of it) is included as the file `blay3d.log`.

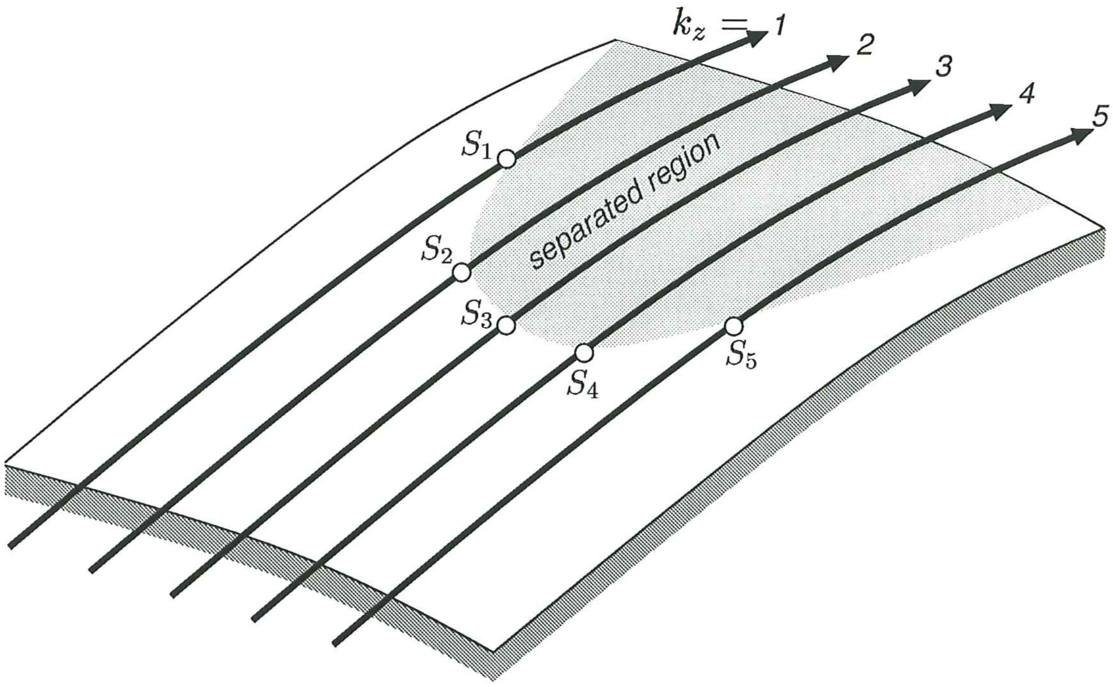


Figure 17: Streamlines may separate individually and computation is continued for the other ones.

6.1 The flat plate

(CDL file `blasius.cdl`). This is the well known case of a flat plate at null incidence angle with respect to a homogeneous flow. The flow is not perturbed by the plate, and then the external velocity field is simply U_{ext} =cnst. The geometry and velocity field are built-in as `icase=2`. The plate is the half plane $y = 0$, $x > 0$, and the velocity is aligned with the $+x$ direction. As the geometry spans indefinitely in the z direction, we take 4 streamlines starting at $x = y = 0$, z =0, 0.1, 0.2 and 0.3. (We could take also three streamlines, but not less, because of the restriction discussed at the last paragraph of §4.7). The direction of start $\hat{\mathbf{e}}_1$ is always in the x direction.

After interpretation of the variables, the following output is produced

```
 Similar flow, (phi prop. to cu*x**m), m,cu:  0.  1.
============================================
Total area integrated :                                    .59910000E+01
Total force :                  .50515865E-02  .00000000E+00  .00000000E+00
Total torque :                 .00000000E+00 -.75773798E-03  .00000000E+00
```

The first line indicates that BLAY3D has properly identified the similarity expansion at the LE: $U_{\text{ext}} = 1$. After the specified number of computations (the results are in the Fortran unit 10), BLAY3D prints the total force and torque computed. The kinematic

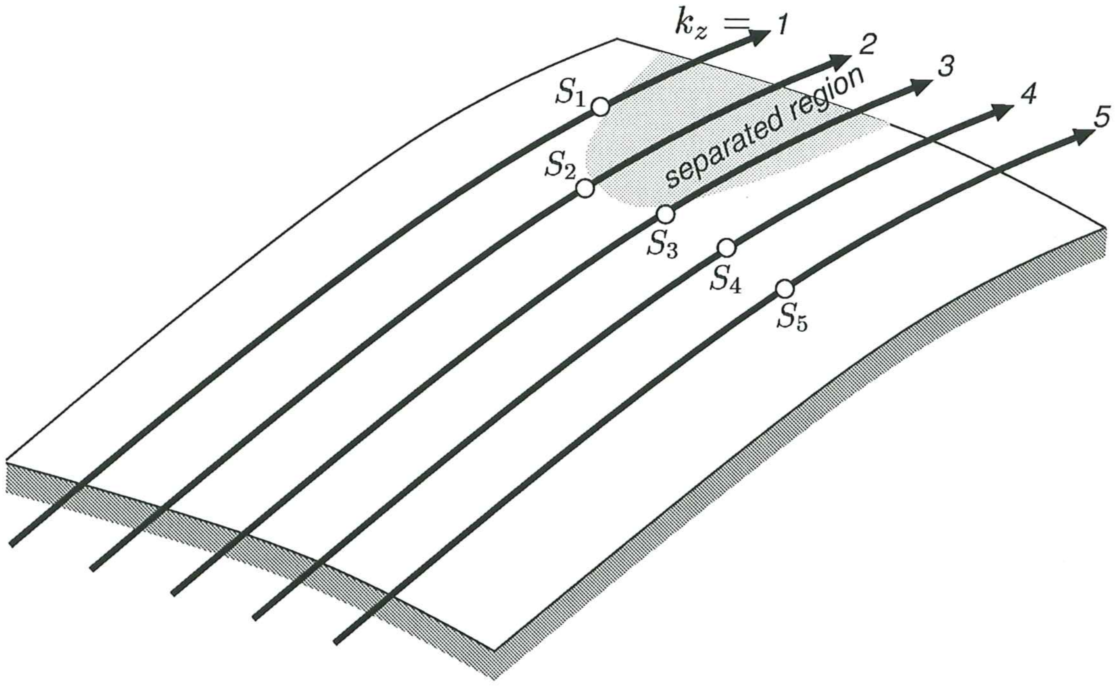


Figure 18: Streamlines 4 nd 5 marked as separated because of not enough neighbor ones.

viscosity is $\nu = 10^{-7}$, and computations end at $L = 20$ so that $\mathrm{Re}_L = 2\times10^2$ and the flow is fully turbulent in most part of the plate. The total force is in accordance to experimental results [12]. Some negative torque is reported around the y axis, since the domain of integration is on the $z > 0$ part.

6.2 Sphere

(CDL file `sphere.cdl`). The sphere has unit radius and is located at $(x, y, z) = (0, 0, 0)$. The unperturbed velocity of magnitude $U_\infty = 1$ is aligned with the positive $+x$-axis. The inviscid velocity field is (on the surface of the sphere) given by

$$u = 1.5\,(y^2 + z^2) \tag{13}$$
$$v = -1.5\,xy \tag{14}$$
$$w = -1.5\,xz \tag{15}$$

The geometry and velocity field are built-in as `icase=3`. This is the potential solution assuming no separation of the boundary layer. Off course, this is not true, the boundary layer separates, and this affects the potential velocity in such a way that the boundary layer solution is not applicable. However, this case has been taken as a benchmark and results are found profusely in the literature so that we include it as a benchmark problem.

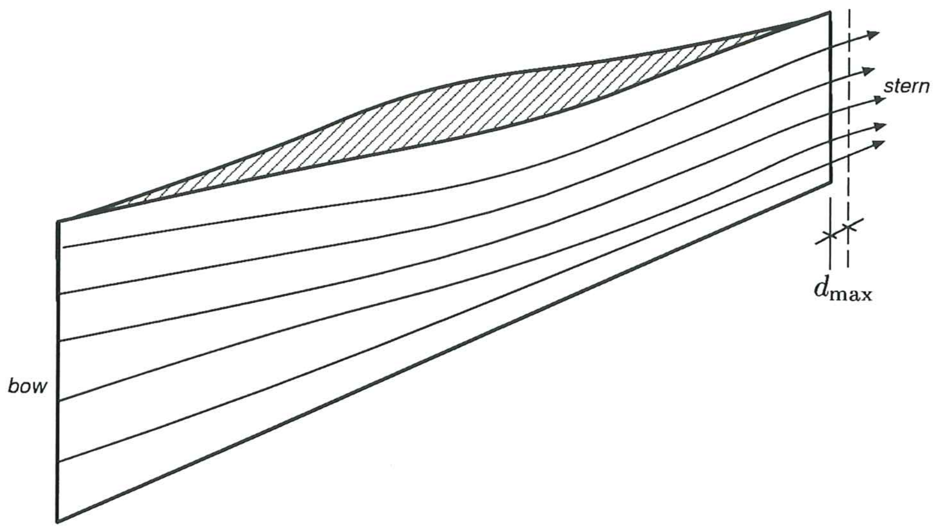


Figure 19: Stopping calculations due to streamlines exit from the surface.

This is the typical case of a stagnation point and the we set `initial_axf` to 1. We take 11 streamlines starting at the stagnation region $(x, y, z) = (-1, 0, 0)$, and with directions at angles linearly interpolated between $(\hat{\mathbf{e}}_1)_1 = (0, 1, 0)$ and $(\hat{\mathbf{e}}_1)_{11} = (0, 1, 1)/\sqrt{2}$. This represents an aperture of 45° so that there are 4.5° between streamlines.

As the flow is axially symmetric about the x-axis, the streamlines are meridians and we could have used equally well any other aperture angle, or the whole sphere. The flow separates near at an angle of 106.2°, which is in agreement with data reported in the literature.

6.3 Circular cylinder

(CDL file `sphere.cdl`). The cylinder has unit radius with the z-axis as its axis. The unperturbed velocity field is $(1, 0, 0)$, and the inviscid velocity field (on the surface) is given by

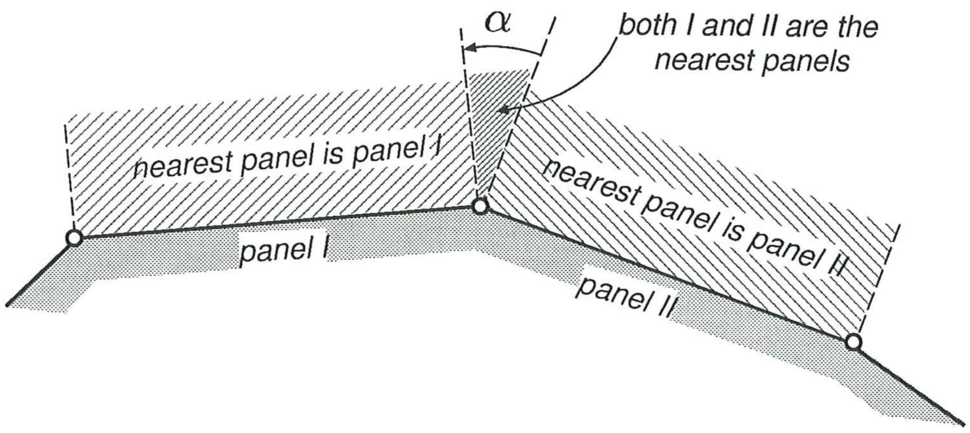
$$u = 2\,y^2 \tag{16}$$

$$v = -2xy \tag{17}$$

The geometry and velocity field are built-in as `icase=4`. Four streamlines are started at the stagnation line $(-1, 0, z)$ with z interpolated linearly between 0 and 0.03, and directions $(\hat{\mathbf{e}}_1)_k = (0, 1, 0)$ for all k. The flow separates at 104.90° in accordance with the literature.

6.4 Prolate ellipsoid

(CDL files in directory `/ELLIPSOID`) This is a prolate ellipsoid whose axis are in the ratio 2.5:1:1. The largest one is aligned with the x-axis, and we consider non-perturbed



α
both I and II are the nearest panels
nearest panel is panel I
nearest panel is panel II
panel I
panel II

Figure 20: Nearest panel to a point

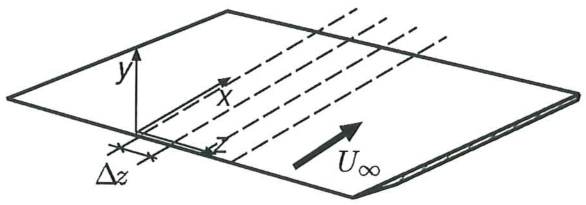


Figure 21: Geometrical setup for the flat plate problem.

velocities $\mathbf{u}_\infty$ aligned or almost aligned with it, i.e.

$$\mathbf{u}_\infty = (\cos\alpha, \sin\alpha\,\cos\theta, \sin\alpha\,\sin\theta) \tag{18}$$

with α small. Off course, as the ellipsoid is symmetric under rotations about the x-axis, the results are independent of the attitude angle θ.

The CDL file `ellip.cdl` corresponds to flow aligned with the axis of the ellipsoid. It reads nodes coordinates, topologies and velocities from the files `el.xyz`, `ellip.top` and `elar-2.5.velx` respectively. Note that changing the angles of attack and attitude α and θ only modify the velocity file. Velocity fields are also supplied for unperturbed velocities aligned with the y (`elar-2.5.vely`) and z (`elar-2.5.velz`) axis, so that by superposition the user can reconstruct the velocity field for any α and θ by computing

$$\mathbf{u}_{\alpha,\theta} = \cos\alpha\,\mathbf{u}_x + \sin\alpha\,\cos\theta\,\mathbf{u}_y + \sin\alpha\,\sin\theta\,\mathbf{u}_z \tag{19}$$

This has to be done with an exterior program, be stored in a file say `ellip.vel` and modify the corresponding line in the CDL file. Except for the case $\alpha = 0$ we have always to set `auto_stag=1` in order to automatically detect the stagnation point.

A file with coordinates for an ellipse with axis in the ratio 5:1:1 (`elar-5.xyz`) is also provided and velocities along the x and y axis (`elar5.velx` and `elar5.vely`). In the case

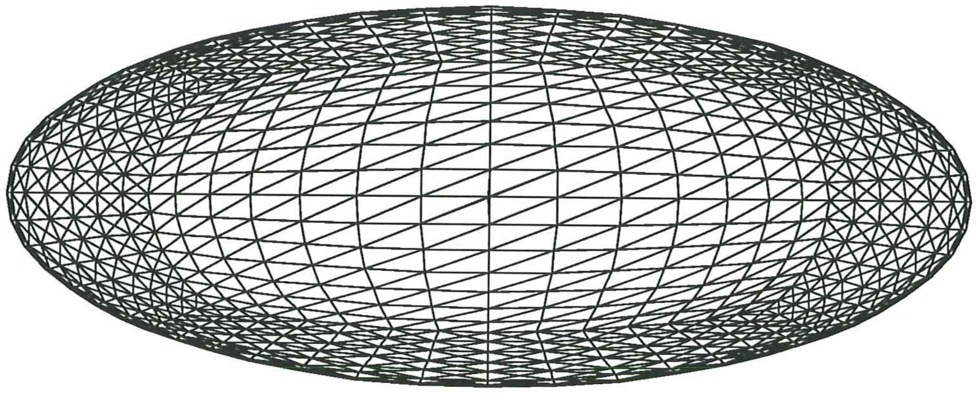


Figure 22: Prolate ellipsoid axis 2.5:1:1

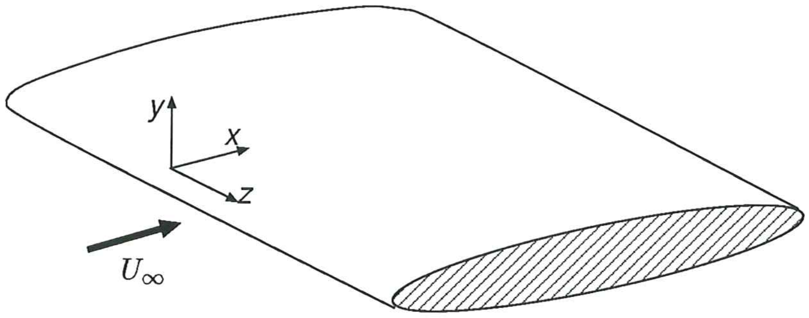


Figure 23: Elliptical cylinder of axis ratio 5:1

of velocity aligned with the x-axis ($\alpha = 0$), the flow is axially symmetric. The velocity vector is always aligned with the meridians and the absolute value of velocity u is only a function of x. A good representation can be obtained by fitting u versus $\xi = \sqrt{1-x^2}$, and we found that

$$u(\xi) = (p_1\,\xi^4 + p_2\,\xi^3 + p_3\,\xi^2 + p_4\,\xi) \tag{20}$$

$$p_1 = -4.2971 \tag{21}$$

$$p_2 = 11.8373 \tag{22}$$

$$p_3 = -12.1312 \tag{23}$$

$$p_4 = 5.6425 \tag{24}$$

gives a satisfactory representation. This is the built-in case `icase=5`.

6.5 Elliptical cylinder

This is an elliptical cylinder of ratio 5:1 that spans in the z direction. The unperturbed velocity field is parallel to the x-axis and then, the velocity field has null z component

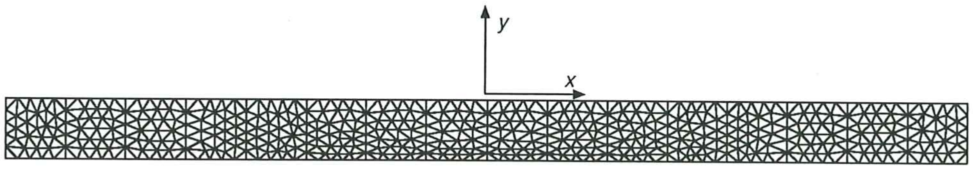


Figure 24: Wigley hull model 1805 A

obtained by a panel method and adjusted with a 4th order polynomial like in §6.4. The coefficients of the adjustment are

$$u(\xi) = (p_1\,\xi^5 + p_2\,\xi^4 + p_3\,\xi^3 + p_4\,\xi^2 + p_5\,\xi) \tag{25}$$

$$p_1 = 1.9676 \tag{26}$$
$$p_2 = -9.5333 \tag{27}$$
$$p_3 = 17.1917 \tag{28}$$
$$p_4 = -14.9028 \tag{29}$$
$$p_5 = 6.4520 \tag{30}$$

This is coded as the built-in case `icase=6`.

6.6 Wigley hull

(CDL file `wigley.cdl`) This is well known ship used as a benchmark for numerical codes. The hull is very thin, it has a draft of 1 and a length of 16. The plane of symmetry is $y = 0$ and the projection of the hull on the $y = 0$ axis occupies the region $-1 < z < 0$, $|x| < 8$. As the leading edge is very sharp (like a flat plate) we know the position of the stagnation line and set `auto_stag` to 0. Then we use 31 streamlines starting at $y = 0$, $x = -8$ and z ranging from $z = 0$ to -1. We interpolate them with `interp` set to 1. As it is very likely that streamlines do not separate at the TE we set a rather low value of `max_in_plane_distance` (0.05).

7 Symbols and Acronyms

7.1 Acronyms

BLAY3D: Boundary LAYer 3D, the code described in this guide.

BLE: Boundary Layer Equations

CDL: network Common Data form Language

FDM: Finite Difference Method

LE: Leading Edge

NetCDF: Unidata Network Common Data Form

NSE: Navier Stokes Equations

ODE: Ordinary Differential Equations

8 Symbols

- $u, v =$ Streamwise and normal components of velocity.
- $\nu =$ Kinematic viscosity.
- $U_{\text{ext}} =$ The (inviscid) velocity outside the boundary layer.

References

[1] Pruett C. On the wall-normal velocity of the compressible boundary layer equations. Technical Report NASA-CR 4419, NASA, 1991.

[2] Pruett C.D. and Streett C.L. A spectral collocation method for compressible, non-similar boundary layers. *International Journal for Numerical Methods in Fluids*, 13:713–737, 1991.

[3] Gottlieb D. and Orszag S.A. *Numerical Analysis Of Spectral Methods: Theory And Applications.* SIAM, 1977.

[4] Hytopoulos E., Schetz J.A., and Gunzburger M. Numerical solution of the compressible boundary layer using the finite element method. Technical Report 92-0666, AIAA, 1992.

[5] Oñate E., Idelsohn S.R., Zienkiewicz O.C., and Taylor R.L. A finite point method in computational mechanics. applications to convective trnsport and fluid flow. *International Journal for Numerical Methods in Engineering*, 39(22):3839–3886, 1996.

[6] Oñate E., Idelsohn S.R., Zienkiewicz O.C., Taylor R.L., and Sacco C. A stabilized finite point method for analysis of fluid mechanics problems. *Computer Methods in Applied Mechanics and Engineering*, 1996.

[7] White F.M. *Viscous Fluid Flow.* McGraw-Hill, New York, 1974.

[8] Schlichting H. *Boundary Layer Theory.* Pergamon Press, 1955.

[9] Schetz J.A. *Foundation of Boundary-Layer Theory for Momentum, Heat and mass Transfer.* Prentice Hall, Englewood Cliffs, 1984.

[10] Schetz J.A. Numerical solution of the boundary layer equations using the finite element method. In M.N. Dhaubhadel et.al., editor, *Advances in Finite Element Analysis in Fluid Dynamics*, volume 123. ASME, 1991.

[11] Storti M. The incompressible boundary layer equations in tensor form (in spanish). Technical report, CIMEC - Centro Internacional de Métodos Computacionales en ingeniería, Santa Fe, Argentina, 1997.

[12] Storti M. Casos test para el programa de capa límite 3d: Blay3d. Technical report, CIMEC - Centro Internacional de Métodos Computacionales en ingeniería, Santa Fe, Argentina, 1998.

[13] Voigt R.G., Gottlieb D., and Hussaini M.Y. *Spectral Methods For Partial Differential Equations.* SIAM, 1984.

[14] Zang T.A., Streett C., and Hussaini M.Y. Spectral methods for cfd. Technical Report 89-13, ICASE, 1989.

[15] Wie Y.S. and Harris J.E. Numerical solution of the boundary layer equations for a general aviation fuselage. *Journal of Aircraft*, 28(12):861–868, 1991.

Index