Práctica 3. Cálculo de un perfil aerodinámico

E. Martín[1], M. Meis[1,2] y F. Varas[1]

[1]Univ. de Vigo, [2]Vicus Desarrollos Tecnológicos

Dinámica de fluidos computacional con OpenFOAM
18–20 de Junio de 2014

PROGRAMA
COOPERACIÓN TRANSFRONTERIZA
ESPAÑA - PORTUGAL
COOPERAÇÃO TRANSFRONTEIRIÇA
2007 - 2013

CloudPYME

Unión Europea
FEDER
Invertimos en su futuro

Proyecto CloudPYME

El proyecto CloudPYME (ID 0682_CLOUDPYME2_1_E) está cofinanciado por la Comisión Europea a través del Fondo Europeo de Desarrollo Regional (FEDER), dentro de la tercera convocatoria de proyectos del Programa Operativo de Cooperación Transfronteriza España–Portugal 2007–2013 (POCTEP).

Outline

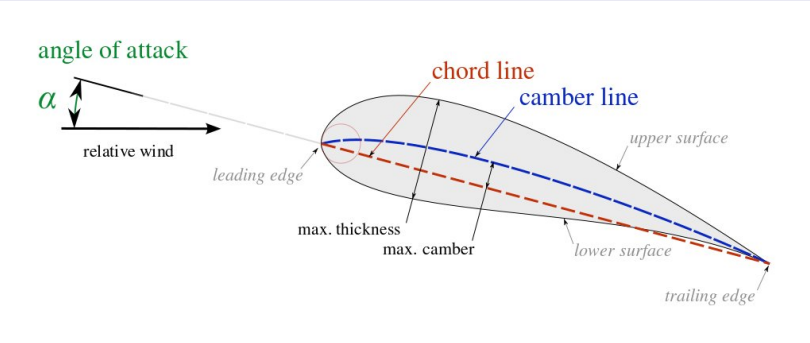
Plan

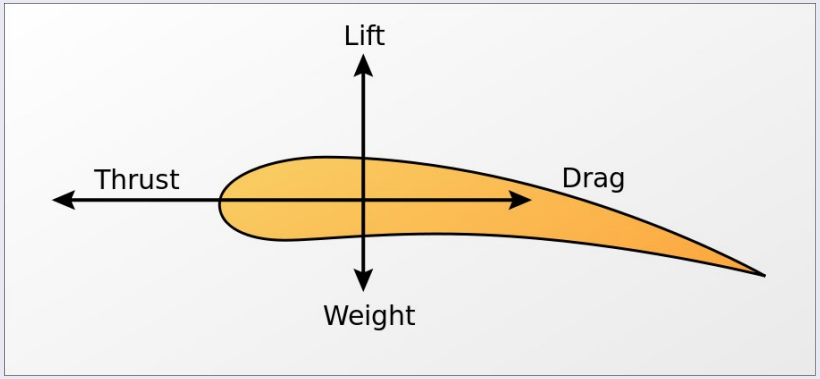
Plan

Diseño de perfiles aerodinámicos

Sustentación generada por perfiles

- cara de presión / cara de succión (intradós / extradós)
- resultante neta: sustentación + arrastre
- dependiente de velocidad y ángulo de ataque



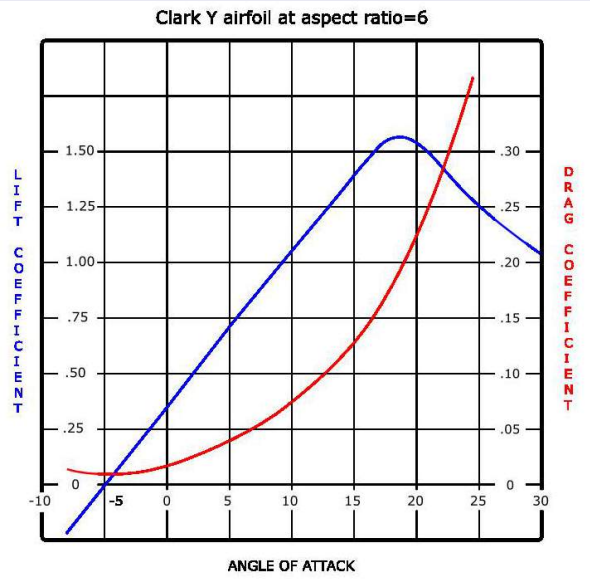


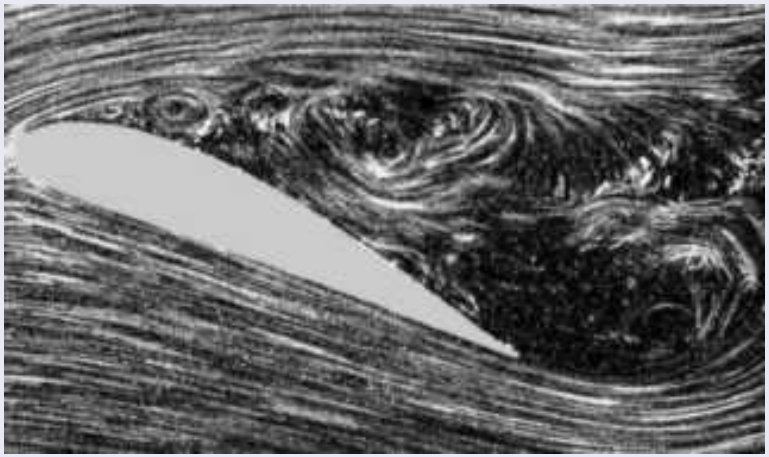
bordes de ataque y fuga / cuerda y línea de curvatura media

Sustentación generada por perfil

Sustentación en función de ángulo de ataque (AoA)

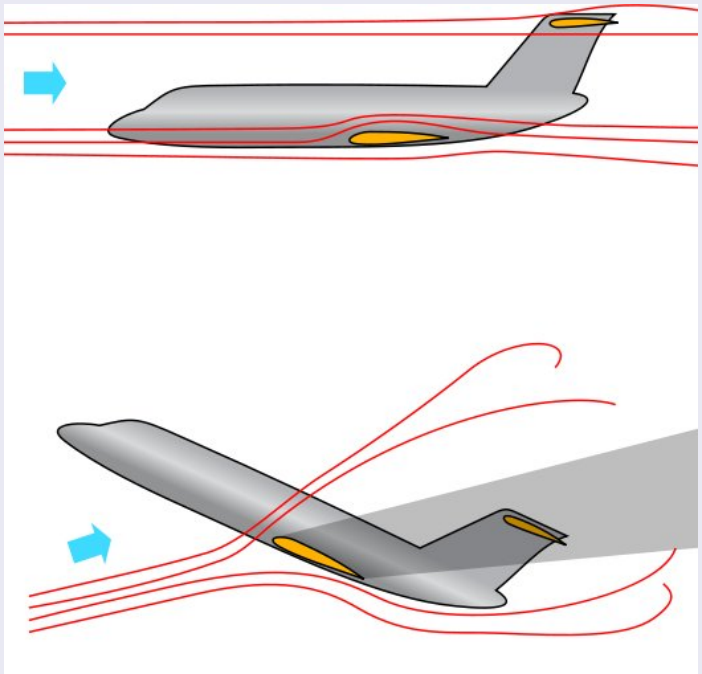
- alternativas de cálculo para pequeños AoA
- desprendimiento de capa para altos AoA

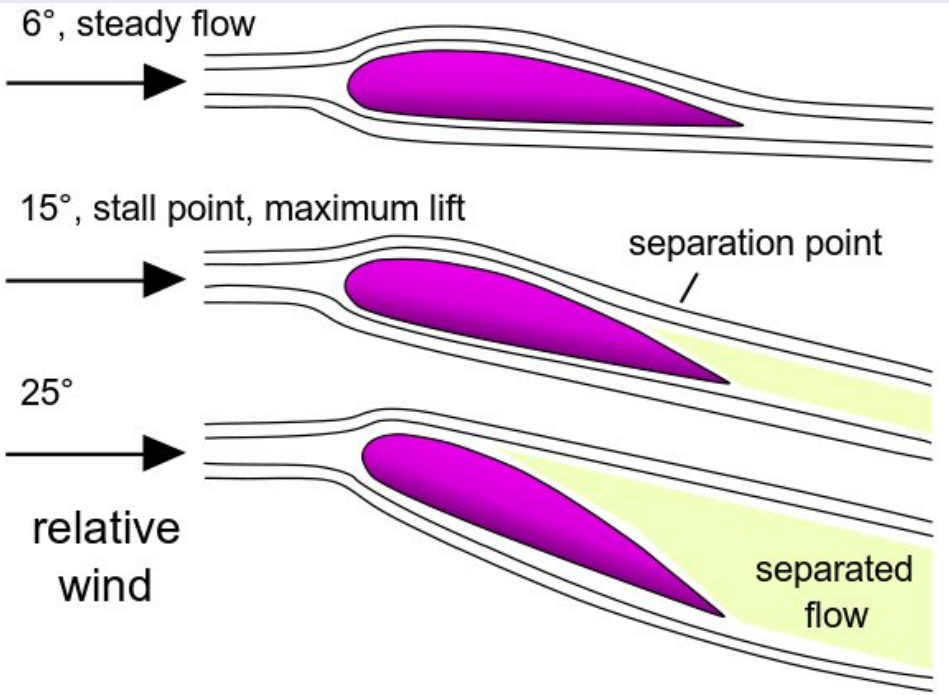




Sustentación y entrada en pérdida

Entrada en pérdida

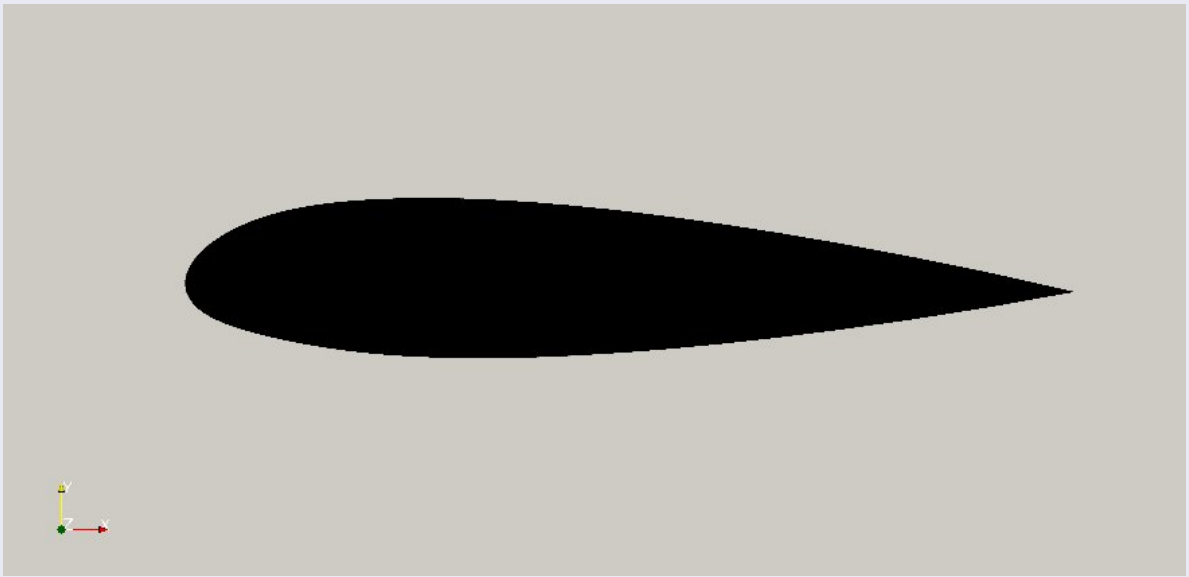




Formulación del problema

Cálculo de sustentación de perfil

- ángulo de ataque: 8 grados
- velocidad: 26 m/s



- perfil (figura) con 9 m de cuerda

Plan

Modelado del flujo

Flujo turbulento

- número de Reynolds en torno a 1.5×10^7
- modelo RANS (Reynolds Averaged Navier–Stokes) estacionario
- modelo turbulencia de Spalart–Almaras

Spalart–Almaras

- Modelo de turbulencia de una ecuación

$$\mu_t = \rho \tilde{\nu} f_{\nu 1}(\frac{\tilde{\nu}}{\nu})$$

$$\frac{\partial \tilde{\nu}}{\partial t} + \vec{U} \cdot \vec{\nabla} \tilde{\nu} = \text{Difusión} + \text{Producción} - \text{Destrucción}$$

- modelo propuesto para flujos en torno a perfiles

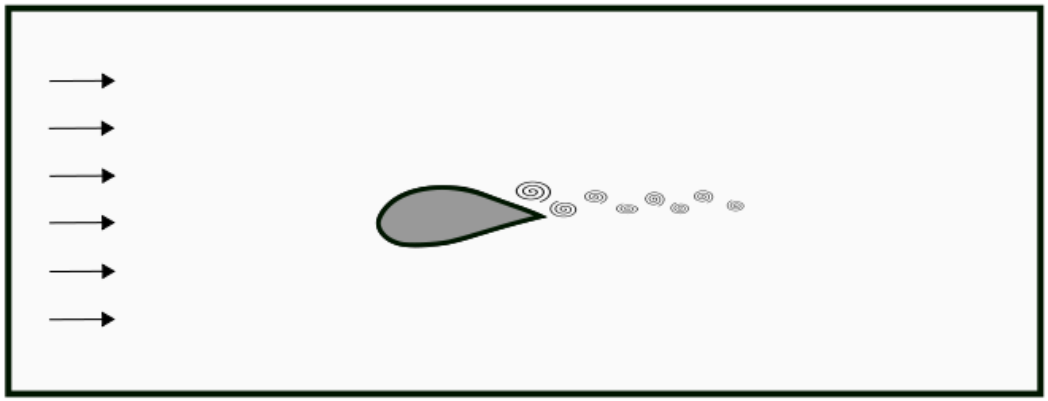
Condiciones de contorno

Sobre paredes del perfil

Leyes de pared (evita resolver capa límite)

Sobre fronteras artificiales

- flujo no perturbado en entrada
- condiciones de salida *neutras*



Modelo numérico

Esquema de resolución

Algoritmo SIMPLE (problema estacionario)
Principales parámetros a fijar:

- número de iteraciones
- correcciones no ortogonales
- parámetros de relajación

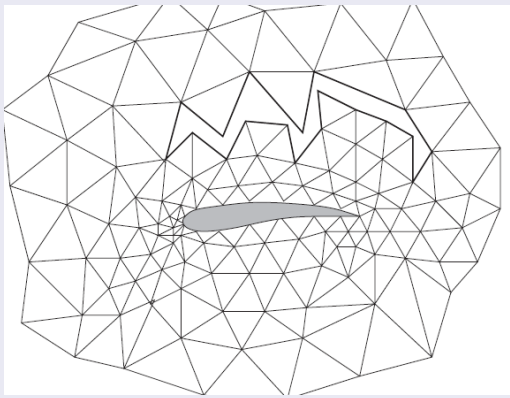
Aproximación de flujos

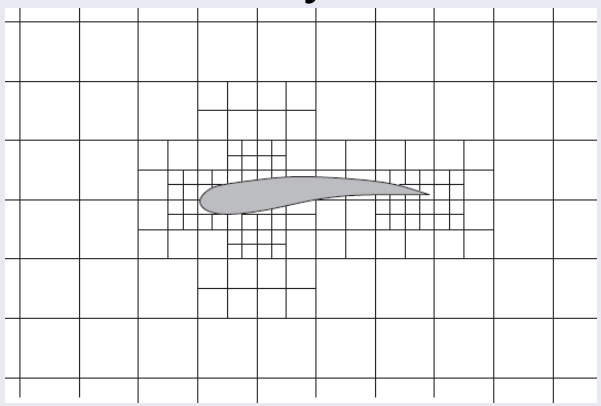
- aproximación de terminos convectivos (descentrado)
- aproximación de términos *difusivos*

Modelo numérico (cont.)

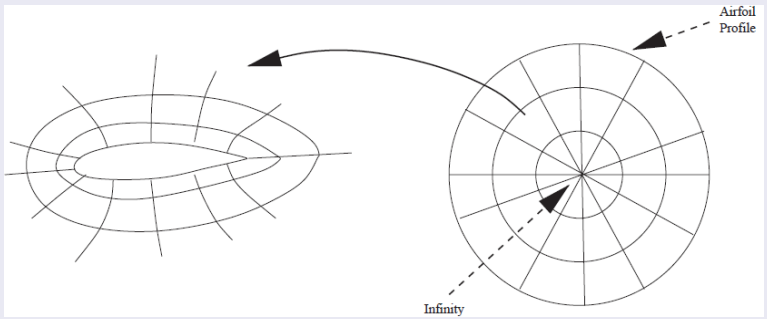
Generación de mallados (Baker, Progr. Aerospace Sci. 2005)

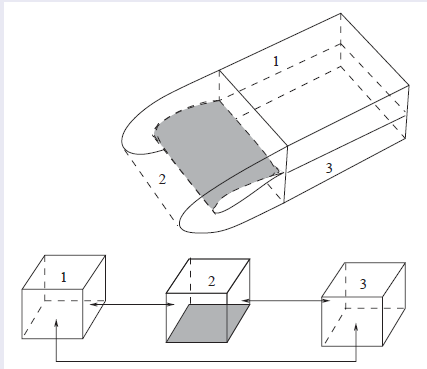
mallas no estructuradas: frontal y OctTree





mallas estructuradas: O–mesh y C–mesh





Plan

Plan

Obtención de mallado

Importación de mallado

OpenFOAM puede convertir mallas de:

- Gmsh (SW libre, con lenguaje propio)
- Gambit/Fluent
- IDEAS, ANSYS ...

`www.openfoam.com/features/mesh-conversion.php`

Generación manual de mallado estructurado

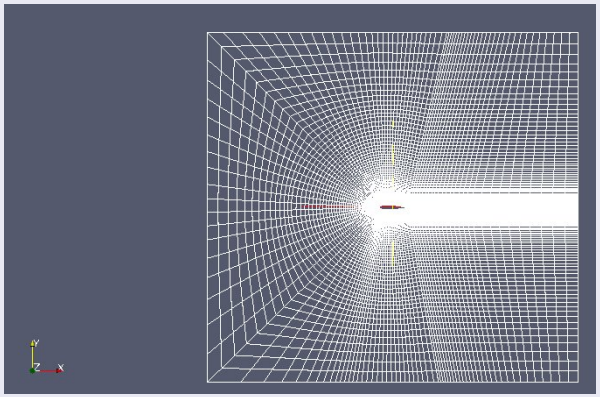
Código (sencillo) para generar

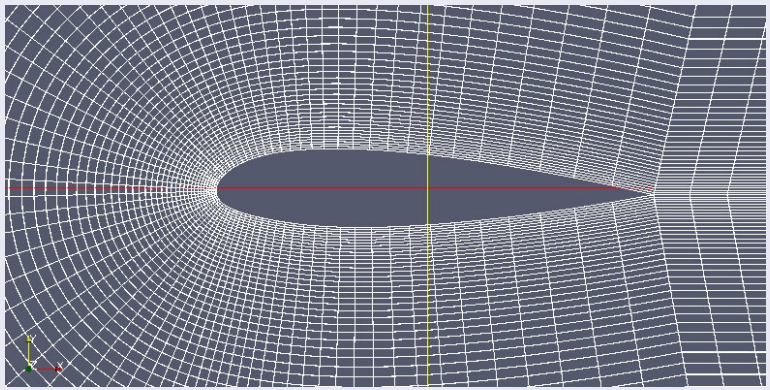
- O–malla
- C–malla

y almacenar en formato OpenFOAM (descrito en manuales)

Mallado disponible en tutorial

Malla en C





Visualización de la malla usando Paraview

1. Convertir a VTK (desde directorio del caso)

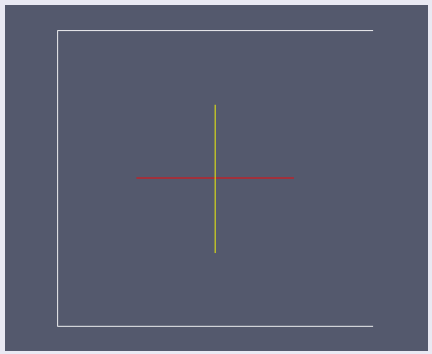
```
$ foamToVTK
```

2. Abrir `airfoil2D` (en directorio `VTK`) y usar `WireFrame`

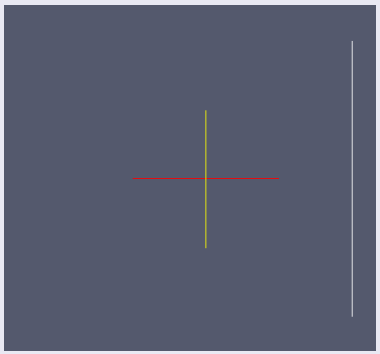
Mallado disponible en tutorial (cont.)

Identificación de fronteras con Paraview

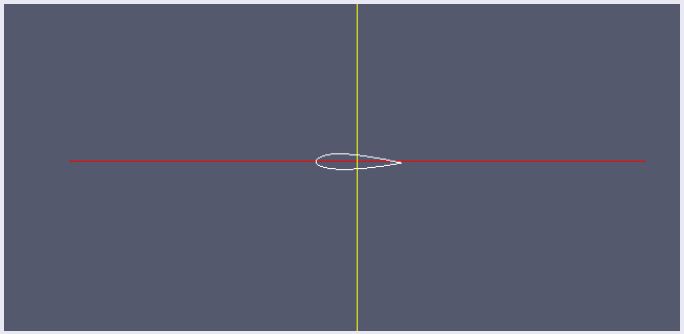
Abrir componentes `inlet`, `outlet` y `wall` (en directorio `VTK`)



`inlet`



`outlet`



`wall`

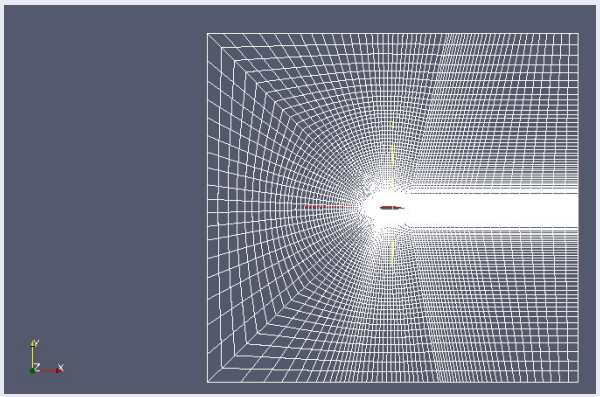
Existen otras alternativas:

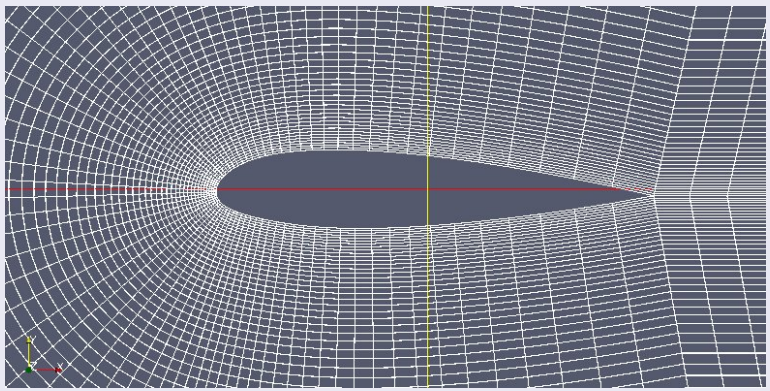
- ParaFoam (etiqueta fronteras)
- convertir a formato OBJ

Mallado disponible en tutorial (cont.)

Observaciones sobre mallado

- buenas propiedades de ortogonalidad
- (solo) adecuado para bajos ángulo de ataque
- (relativamente) baja resolución





Plan

Implementación de modelo físico

Descripción del modelo físico

- geometría (y mallado)
- modelo de turbulencia
- propiedades del fluido
- condiciones de contorno

Implementación de modelo físico

Directorio `constant`

- subdirectorio `polyMesh` (geometría/mallado)
- Diccionario `RASProperties` (turbulencia)
- Diccionario `transportProperties` (fluido)

Diccionario `RASProperties`

```
RASModel    SpalartAllmaras;
turbulence    on;
```

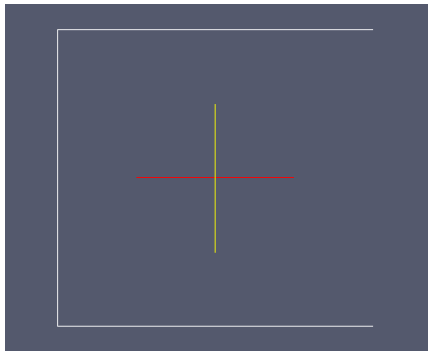
Diccionario `transportProperties`

```
transportModel     Newtonian;
rho      rho [ 1 -3 0 0 0 0 0 ] 1;
nu       nu [ 0 2 -1 0 0 0 0 ] 1e-05;
```

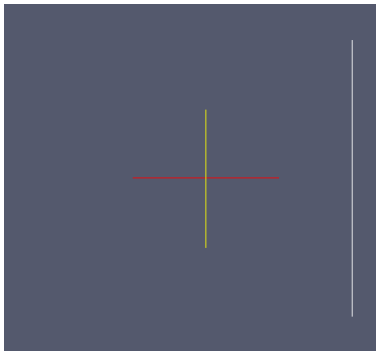
Implementación de modelo físico

Condiciones de contorno (e iniciales)

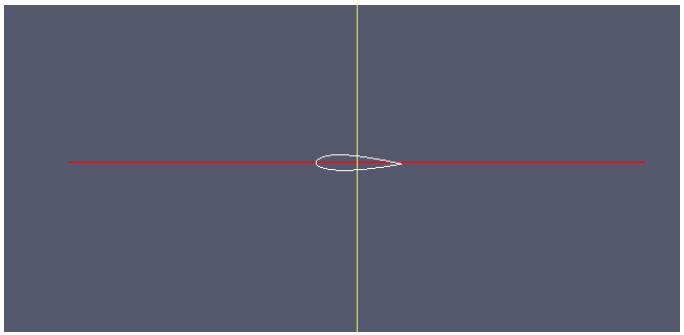
velocidad: archivo `0/U`
presión: archivo `0/p`
variable $\tilde{\nu}$: archivo `0/nuTilda`



`inlet`



`outlet`



`wall`

Implementación de modelo físico

Condiciones sobre velocidad

- en frontera `inlet`: $\vec{U} = (U_\infty \cos\alpha, U_\infty \mathrm{sen}\alpha)$
- en frontera `wall`: ley de pared
- en frontera `outlet`: condición *neutra*

Observaciones

- se fijan condiciones iniciales (esquema SIMPLE)
- en ley de pared:
  - la frontera debe tener tipo adecuado (`wall`)
  - se fija velocidad de la pared
- en condiciones *neutras* se puede usar:
  - gradiente nulo
  - gradiente nulo + tratamiento de recirculación

Implementación de modelo físico

Tipos de frontera: archivo `constant/polyMesh/boundary`

```
inlet
{
    type patch;
    physicalType inlet;
    nFaces 134;
    startFace 21254;
}

outlet
{
    type patch;
    physicalType outlet;
    nFaces 160;
    startFace 21388;
}

wall
{
    type wall;
    physicalType wall;
    nFaces 78;
    startFace 21548;
}

frontAndBack
{
    type empty;
    physicalType empty;
    nFaces 21440;
    startFace 21626;
}
```

Implementación de modelo físico

Condiciones sobre velocidad: archivo `0/U`

```
dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (25.75 3.62 0);

boundaryField
{
    inlet
    {
        type            freestream;
        freestreamValue uniform (25.75 3.62 0);
    }

    outlet
    {
        type            freestream;
        freestreamValue uniform (25.75 3.62 0);
    }

    wall
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    frontAndBack
    {
        type            empty;
    }
}
```

Implementación de modelo físico

Condiciones sobre presión: archivo `0/p`

```
dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            freestreamPressure;
    }

    outlet
    {
        type            freestreamPressure;
    }

    wall
    {
        type            zeroGradient;
    }

    frontAndBack
    {
        type            empty;
    }
}
```

Implementación de modelo físico

Condiciones sobre variable $\tilde{\nu}$: archivo `0/nuTilda`

```
dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0.14;

boundaryField
{
    inlet
    {
        type            freestream;
        freestreamValue uniform 0.14;
    }

    outlet
    {
        type            freestream;
        freestreamValue uniform 0.14;
    }

    wall
    {
        type            fixedValue;
        value           uniform 0;
    }

    frontAndBack
    {
        type            empty;
    }
```

Implementación de modelo numérico

Selección de modelo numérico

- Algoritmo SIMPLE (con parámetros asociados)
- Discretización de flujos
- Resolución de sistemas lineales

Información en directorio `system`

- Diccionario `controlDict`
- Diccionario `fvSchemes`
- Diccionario `fvSolution`

Implementación de modelo numérico

Algoritmo numérico en `system/controlDict`

```
application     simpleFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         200;

deltaT          1;

writeControl    timeStep;

writeInterval   50;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;
```

Implementación de modelo numérico

Aproximación de flujos en `system/fvSchemes`

```
ddtSchemes
{
    default         steadyState;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      bounded Gauss linearUpwind grad(U);
    div(phi,nuTilda) bounded Gauss linearUpwind grad(nuTilda);
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}
```

Implementación de modelo numérico

Resolución de sistemas lineales en `system/fvSolution`

```
solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0.1;
        smoother        GaussSeidel;
        nPreSweeps      0;
        nPostSweeps     2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
    }

    U
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        nSweeps         2;
        tolerance       1e-08;
        relTol          0.1;
    }

    nuTilda
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        nSweeps         2;
        tolerance       1e-08;
        relTol          0.1;
    }
```

Implementación de modelo numérico

Parámetros de SIMPLE en `system/fvSolution`

```
SIMPLE
{
    nNonOrthogonalCcorrectors 0;
    pRefCell        0;
    pRefValue       0;
}

relaxationFactors
{
    default         0;
    p               0.3;
    U               0.7;
    nuTilda         0.7;
}
```

Plan

Resolución del caso

Ejecución del caso

Ejecución y almacenamiento de salida en archivo `log`:

```
$ simpleFoam > log
```

Revisión de convergencia del algoritmo SIMPLE

Postprocesamiento de archivo de salida:

```
$ foamLog log
```

Consulta de residuos en ecuación de momentos (en eje *OX*):

```
$ octave
octave:> load logs/UxFinalRes_0
octave:> iter = UxFinalRes_0(:,1)
octave:> UxRes = UxFinalRes_0(:,2)
octave:> plot( iter, UxRes )
```

Control de la convergencia

Archivo `system/fvSolution`

Establecimiento de criterios de parada:

```
SIMPLE
{
    nNonOrthogonalCorrectors 0;
    pRefCell         0;
    pRefValue        0;

    residualControl
    {
        p               1e-5;
        U               1e-5;
        nuTilda         1e-5;
    }
}
```

Postprocesado del caso

Generación de archivos VTK

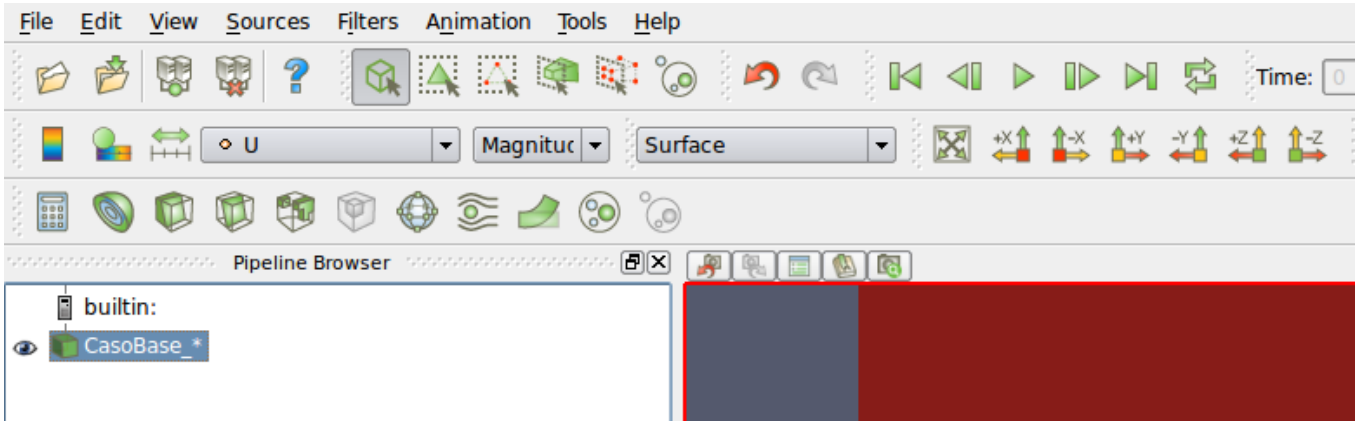
```
$ foamToVTK
```

Visualización con `Paraview`

```
$ paraview
```

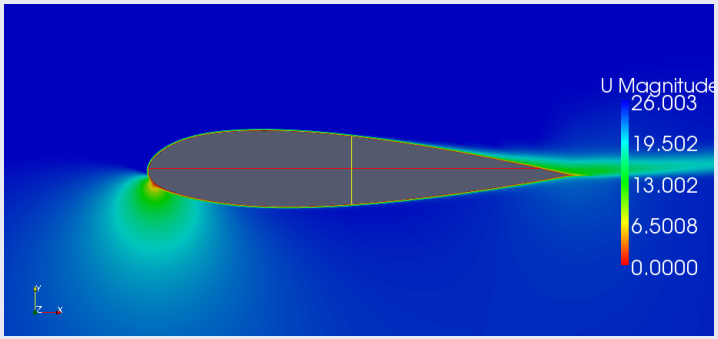
Abrir archivo: `File` > `Open` > `CasoBase_..vtk`

Modificar `Solid Color` por variable a representar

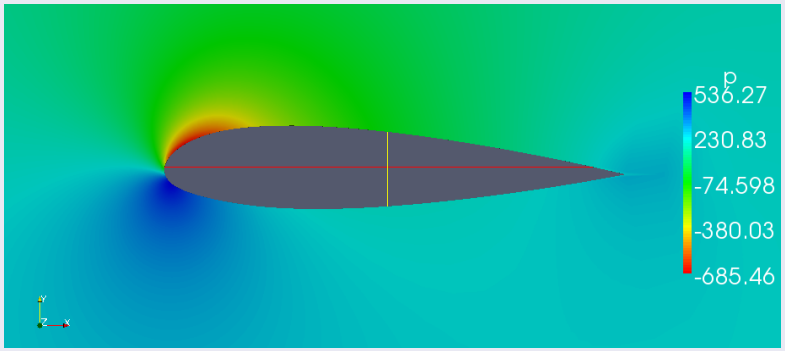


Postprocesado del caso

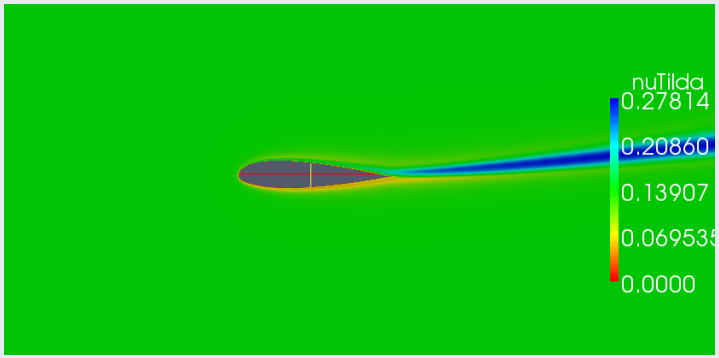
Resultados de simulación



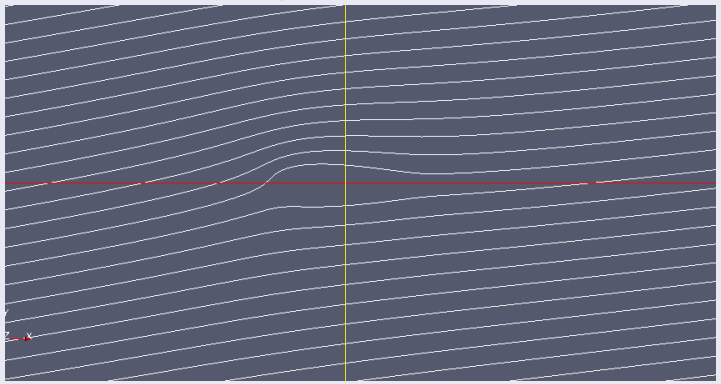
velocidades



presiones



variable $\tilde{\nu}$



líneas de corriente

Plan

Plan

Variación de datos

Variación de velocidad y ángulo de ataque

Se fijan en archivo `0/U`

Un par de posibilidades:

- modificar datos sobre propio archivo

  (se introduce variables para facilitar modificación)
- incluir segundo archivo con datos

  (más sencillo de automatizar)

Modificación sobre archivo

Archivo `0/U`

```
dimensions      [0 1 -1 0 0 0 0];

//  Datos de velocidad: U = 26 m/s , AoA = 15 grados

Ux          25.114;   // componente en X de la velocidad
Uy          6.7293;   // componente en Y de la velocidad

internalField   uniform ($Ux $Uy 0);          // asigna Ux y Uy

boundaryField
{
    inlet
    {
        type            freestream;
        freestreamValue uniform ($Ux $Uy 0);   // asigna Ux y Uy
    }

    outlet
    {
        type            freestream;
        freestreamValue uniform ($Ux $Uy 0);   // asigna Ux y Uy
    }
```

Alternativa variación de datos

Alternativa para modificación de datos

Archivo de datos: `0/include/DatosVelocidad`

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/

Ux              25.114;   // componente en X de la velocidad
Uy              6.7293;   // componente en Y de la velocidad

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

Alternativa variación de datos

Alternativa para modificación de datos

Uso de `include` en archivo `0/U`

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#include "include/DatosVelocidad"

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform ($Ux $Uy 0);

boundaryField
{
    inlet
    {
        type            freestream;
        freestreamValue uniform ($Ux $Uy 0);
    }
```

Resolución del caso

Ejecución del caso

Ejecución y almacenamiento de salida en archivo `log`:

```
$ simpleFoam > log
```

Revisión de convergencia del algoritmo SIMPLE

Postprocesamiento de archivo de salida:

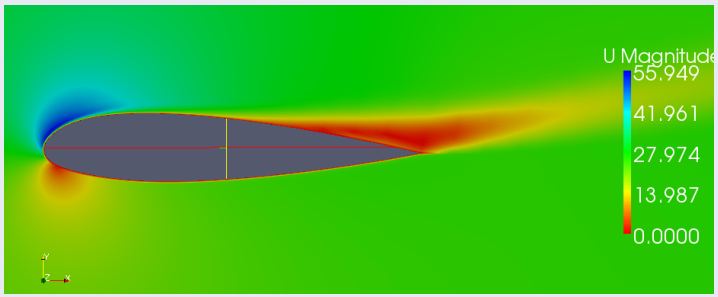
```
$ foamLog log
```

Consulta de residuos en ecuación de momentos (en eje *OX*):

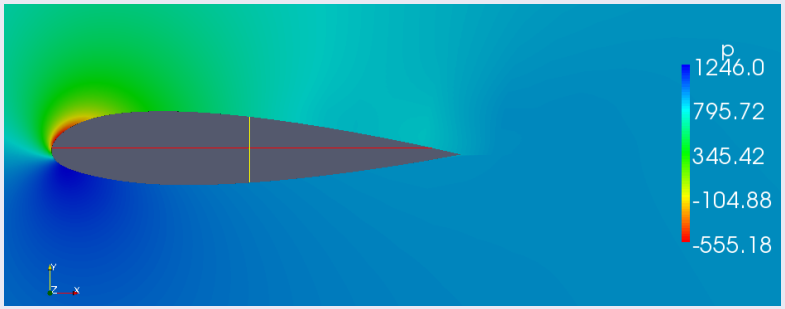
```
$ octave
octave:> load logs/UxFinalRes_0
octave:> iter = UxFinalRes_0(:,1)
octave:> UxRes = UxFinalRes_0(:,2)
octave:> plot( iter, UxRes )
```

Postprocesado del caso

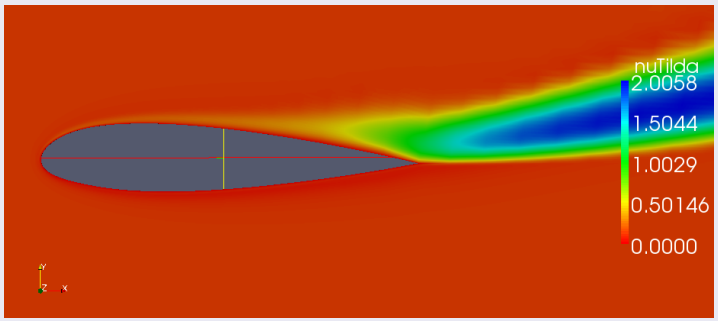
Resultados de simulación



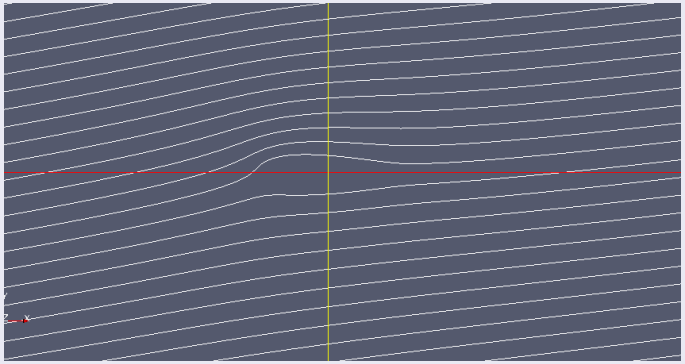
velocidades



presiones



variable $\tilde{\nu}$

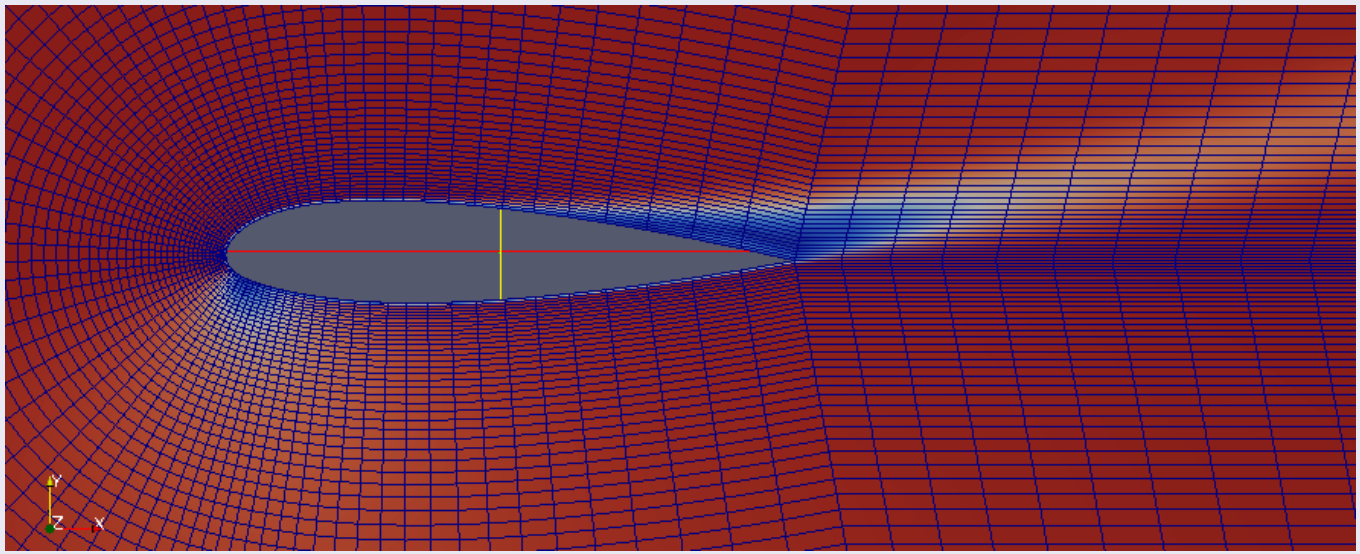


líneas de corriente

Algunas observaciones

Sobre la convergencia

- convergencia más difícil (desprendimiento)
- malla no adaptada a este caso (grosera sobre estela)



Algunas observaciones

Cómo acelerar la convergencia

- usar *continuación* en datos:
  - ángulo de ataque
  - velocidad

Implementación en OpenFOAM

- Sobre misma malla:
  - Usar archivos de variables en cualquier tiempo
- Sobre diferentes mallas:
  - Usar archivos de variables en cualquier tiempo
  - Interpolación entre mallas con `mapFields`

Continuación con OpenFOAM

Archivo `500/U`

Celdas interiores:

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "500";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   nonuniform List<vector>
10720
(
(-3.47396 47.1652 4.29616e-18)
(-0.432773 49.1722 -4.13517e-18)
(3.03959 50.9324 0)
(6.75923 51.8407 -2.82095e-16)
(10.6133 52.0879 2.87942e-16)
(14.469 51.7244 1.66682e-16)
```

Continuación con OpenFOAM

Archivo `500/U`

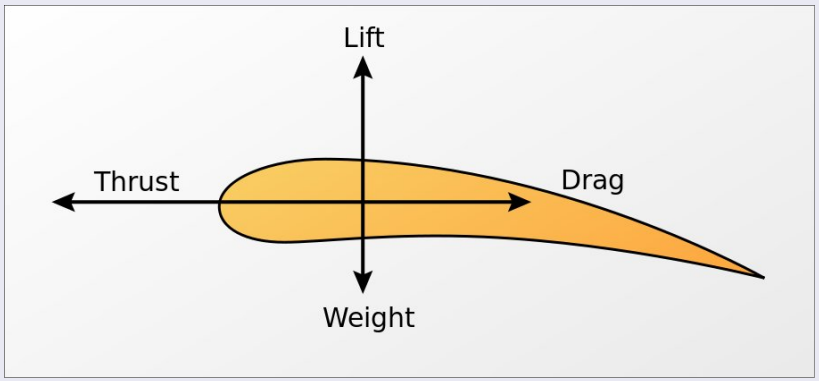
Celdas en frontera de entrada (`inlet`)

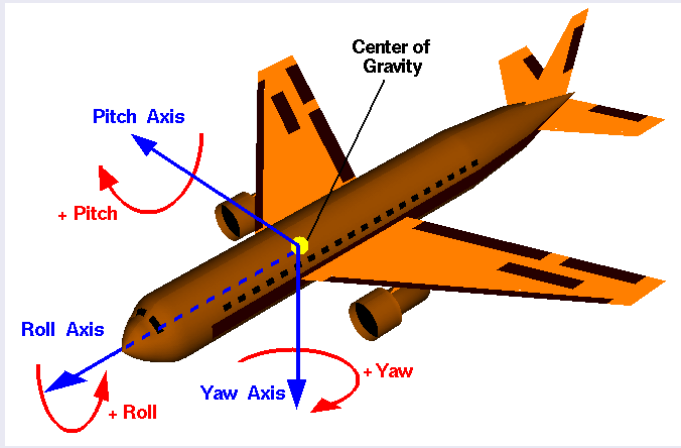
```
boundaryField
{
    inlet
    {
        type            freestream;
        freestreamValue uniform (25.114 6.7293 0);
        value           nonuniform List<vector>
134
(
(25.114 6.7293 0)
(25.114 6.7293 0)
(25.114 6.7293 0)
(25.114 6.7293 0)
(25.114 6.7293 0)
(25.114 6.7293 0)
(25.114 6.7293 0)
(25.114 6.7293 0)
(25.114 6.7293 0)
(25.114 6.7293 0)
(25.114 6.7293 0)
(25.1188 6.73961 -7.99045e-15)
```

Plan

Extracción de esfuerzos

Cálculo de cargas aerodinámicas





Cálculo de:

- sustentación / arrastre
- momento de cabeceo

Extracción de esfuerzos

Modificación de archivo `system/controlDict`

```
timePrecision   6;

runTimeModifiable yes;

libs ( "libOpenFOAM.so" ) ;

functions
(
forces
{
type forces;        // extrae fuerzas ("forces") o coeficientes ("forceCoeffs")
functionObjectLibs ("libforces.so");
patches (wall);     // identifica frontera donde se quieren extraer las fuerzas
pName p;            // variable que almacena la presion
Uname U;            // variable que almacena la velocidad
rhoName rhoInf;     // variable (constante) que almacena la densidad
rhoInf 1.0;         // densidad de referencia
CofR (0 0 0);       // centro para calculo de momentos (Center of Rotation)
outputControl timeStep;
outputInterval 1;
}
);
```

Extracción de esfuerzos

Ejecución del caso

```
$ simpleFoam > log
$ foamLog log
```

Obtención de esfuerzos

Se ejecuta *script* `extract_forces`:

```
$ ./extract_forces
```

Se ejecutan *scripts* de `Octave` para representación:

```
$ octave
octave:> dibuja_sustentacion
octave:> dibuja_esfuerzos
```