Technion - Israel Institute of Technology
Faculty of Aerospace Engineering
Computational Aerodynamics
Project no. 1:
Elliptic Mesh Generation about a NACA 0012 Airfoil Using TTM and ADI

1 Introduction

In this project, you are requested to write a computer program that implements the TTM algorithm and then use it to generate a grid about a two-dimensional airfoil. The program should be capable of generating a "C" type mesh for a NACA 00XX airfoil defined as follows:

$$\begin{aligned} y\left(x\right) = \pm 5 \quad &\cdot \quad t \cdot \left[0.2969\sqrt{x_{int} \cdot x} - 0.1260\left(x_{int} \cdot x\right)\right. \\ &- \quad 0.3516\left(x_{int} \cdot x\right)^2 + 0.2843\left(x_{int} \cdot x\right)^3 \\ &- \quad \left.0.1015\left(x_{int} \cdot x\right)^4\right] \end{aligned} \tag{1}$$

where the plus sign corresponds to the upper surface and the minus sign corresponds to the lower surface, t is the maximum airfoil thickness, and $x_{int} = 1.0$ for the classical NACA 00XX airfoil and $x_{int} = 1.008930411365$ for the modified NACA 00XX airfoil.

The classical NACA 00XX airfoil has a non-closed or blunt trailing edge. The modified NACA 00XX is constructed such that the airfoil exactly closes at the trailing edge. The x and y coordinates and t are normalized by the chord length.

2 Governing Equations

Poisson's equations for the x and y coordinates in curvilinear coordinates are given by:

$$\begin{aligned} \alpha(x_{\xi\xi} + \phi x_{\xi}) - 2\beta x_{\xi\eta} + \gamma(x_{\eta\eta} + \psi x_{\eta}) &= 0 \\ \alpha(y_{\xi\xi} + \phi y_{xi}) - 2\beta y_{\xi\eta} + \gamma(y_{\eta\eta} + \psi y_{\eta}) &= 0 \end{aligned} \tag{2}$$

where

$$\begin{aligned} \alpha &= x_\eta^2 + y_\eta^2 \\ \beta &= x_\xi x_\eta + y_\xi y_\eta \\ \gamma &= x_\xi^2 + y_\xi^2 \end{aligned} \tag{3}$$

The control functions $\phi(\xi,\eta)$ and $\psi(\xi,\eta)$ are calculated using the following relations:

- Boundaries where $\xi = const$

$$\begin{aligned} |y_\eta| > |x_\eta| &\rightarrow \psi(\xi_{min},\eta) = -y_{\eta\eta}/y_\eta|_{\xi=\xi_{min}} \\ |y_\eta| < |x_\eta| &\rightarrow \psi(\xi_{min},\eta) = -x_{\eta\eta}/x_\eta|_{\xi=\xi_{min}} \\ |y_\eta| > |x_\eta| &\rightarrow \psi(\xi_{max},\eta) = -y_{\eta\eta}/y_\eta|_{\xi=\xi_{max}} \\ |y_\eta| < |x_\eta| &\rightarrow \psi(\xi_{max},\eta) = -x_{\eta\eta}/x_\eta|_{\xi=\xi_{max}} \end{aligned} \tag{4}$$

- Boundaries where $\eta = const$

$$\begin{aligned} |x_\xi| > |y_\xi| &\rightarrow \phi(\xi,\eta_{min}) = -x_{\xi\xi}/x_\xi|_{\eta=\eta_{min}} \\ |x_\xi| < |y_\xi| &\rightarrow \phi(\xi,\eta_{min}) = -y_{\xi\xi}/y_\xi|_{\eta=\eta_{min}} \\ |x_\xi| > |y_\xi| &\rightarrow \phi(\xi,\eta_{max}) = -x_{\xi\xi}/x_\xi|_{\eta=\eta_{max}} \\ |x_\xi| < |y_\xi| &\rightarrow \phi(\xi,\eta_{max}) = -y_{\xi\xi}/y_\xi|_{\eta=\eta_{max}} \end{aligned} \tag{5}$$

- The values of ϕ and ψ inside the computational domain are calculated by a linear interpolation.

3 Indexes

From hereto-forth it is assumed that the programming language is "C" and therefore it is assumed that all arrays start from zero. For example, if the number of grid points in the ξ coordinate direction is ni then $\xi_{min} = 0$ and $\xi_{max} = i_{max} = ni - 1$. Similarly, $\eta_{min} = 0$ and $\eta_{max} = j_{max} = nj - 1$ (nj being the number of grid points in the η coordinate direction).

4 Description of the Physical and Computational Domains

Figure 1 contains a description of the physical and computational domains while Figures 2 and 3 contain schematics of the physical and computational domains, respectively.

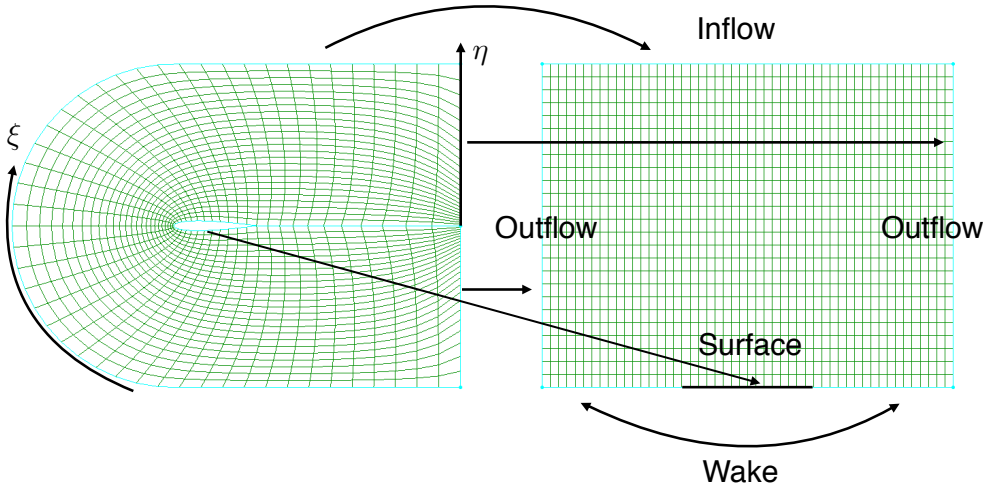

Figure 1: Physical and computational domains

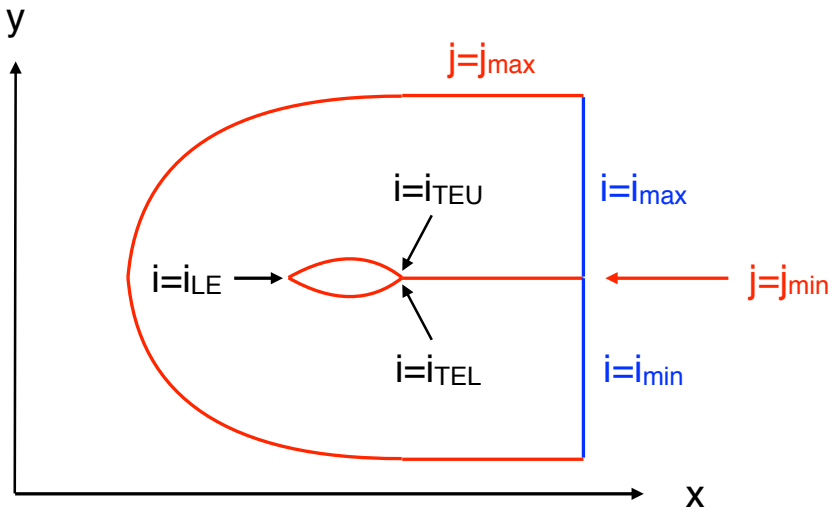

Figure 2: Schematic of the physical domain

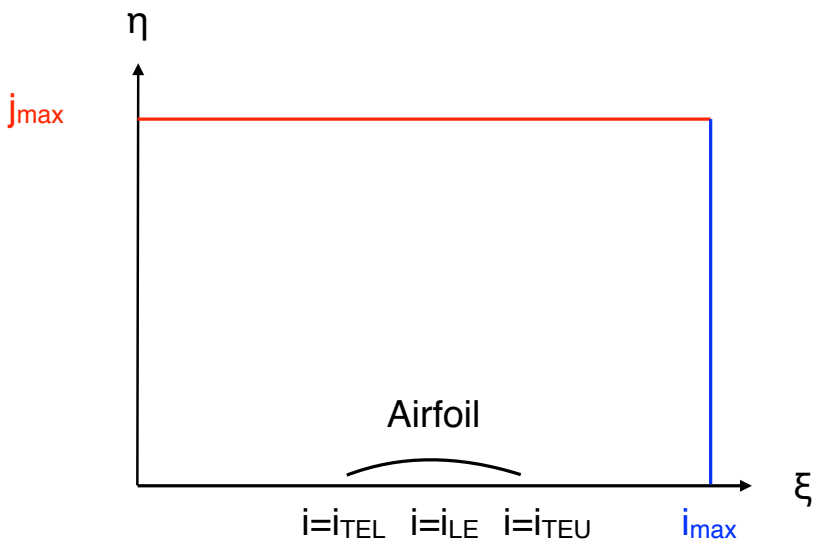

Figure 3: Schematic of the computational domain

5 Boundary and Initial Conditions

Assuming that the leading edge is placed at the origin of the coordinate system, grid points may be distributed along the airfoil surface using:

$$\Delta x = \frac{1.0}{i_{LE} - i_{TEL}}$$

$$\left.\begin{array}{rcc} x_{i,j_{min}} & = & 1 - \cos\left[\frac{\pi(i_{LE}-i)\Delta x}{2}\right] \\ y_{i,j_{min}} & = & y\left(x_{i,j_{min}}\right) \end{array}\right\} \quad i = i_{TEL}, \ldots, i_{LE} \tag{6}$$

The stretching function utilized above allows to stretch the grid points toward the leading edge in order to provide sufficient points to capture the leading edge suction phenomenon. Since the airfoil in question is symmetric, one may find the upper surface grid points, $i = i_{LE}, \ldots, i_{TEU}$, using a mirror transformation (use the "$-$" sign in Equation 1 for y values of the lower surface and the "$+$" sign for the upper surface). The rest of the grid point coordinates for the boundary $j = j_{min}$ may be found using:

$$\left.\begin{array}{rcc} x_{i,j_{min}} & = & x_{i-1,j_{min}} + \left(x_{i-1,j_{min}} - x_{i-2,j_{min}}\right) * XSF \\ y_{i,j_{min}} & = & 0 \end{array}\right\} \quad i = i_{TEU}+1, \ldots, i_{max}$$

$$\left.\begin{array}{rcc} x_{i,j_{min}} & = & x_{i_{max}-i,j_{min}} \\ y_{i,j_{min}} & = & 0 \end{array}\right\} \quad i = i_{min}, \ldots, i_{TEL}-1 \tag{7}$$

where $XSF > 1$ allows stretching of the grid points. The exit-boundary grid points are given by the the following boundary conditions:

$$\begin{aligned} y_{i_{max},j_{min}+1} &= \Delta y \\ y_{i_{max},j} &= y_{i_{max},j-1} + (y_{i_{max},j-1} - y_{i_{max},j-2}) * YSF; \quad j = j_{min}+2, \ldots, j_{max} \\ x_{i_{max},j} &= x_{i_{max},j_{min}}; \quad j = j_{min}+1, \ldots, j_{max} \\ y_{i_{min},j} &= -y_{i_{max},j}; \quad j = j_{min}+1, \ldots, j_{max} \\ x_{i_{min},j} &= x_{i_{min},j_{min}}; \quad j = j_{min}+1, \ldots, j_{max} \end{aligned} \tag{8}$$

The quantities $\Delta y, XSF, YSF, i_{max}$, and j_{max} are case dependent as they should be supplied by the user. The value of Δy is set based on the flow model (inviscid, laminar, or turbulent), XSF and YSF should be less than 1.3 and the dimensions i_{max} and j_{max} are chosen to have a sufficiently long wake and a free stream boundary far enough from the airfoil (at least 5 chords).

On the outer boundary ($j = j_{max}$) use equal spacing to determine the x and y values. Note that this should be arc length and not Δx or Δy individually. The maximum distance to the outer boundary should be set based on the exit boundary. That is, $R_{max} = y_{i_{max},j_{max}}$. The outer boundary should remain parallel to the x axis up to $x = 0$ and then should be closed using a circular arc with the radius R_{max}.

Initial conditions should be established using linear interpolations. You may use the same type of interpolation used to determine the control function inside the computational domain. It is therefore recommended to write a function that does this interpolation (you would need two functions like that for the control functions, one for the ξ direction and one for the η direction.

6 The Solution Scheme

The solution scheme is as follows

$$\begin{aligned} (r - \alpha\delta_{\xi\xi})(r - \gamma\delta_{\eta\eta})C^n_{x_{i,j}} &= rwLx^n_{i,j} \\ (r - \alpha\delta_{\xi\xi})(r - \gamma\delta_{\eta\eta})C^n_{y_{i,j}} &= rwLy^n_{i,j} \\ C^n_{x_{i,j}} &= x^{n+1}_{i,j} - x^n_{i,j} \\ C^n_{y_{i,j}} &= y^{n+1}_{i,j} - y^n_{i,j} \end{aligned} \tag{9}$$

where the operators $Lx^n_{i,j}$ and $Ly^n_{i,j}$ are defined by:

$$\begin{aligned} Lx^n_{i,j} = \alpha^n_{i,j}[(x^n_{i+1,j} - 2x^n_{i,j} + x^n_{i-1,j}) + \frac{\phi_{i,j}}{2}(x^n_{i+1,j} - x^n_{i-1,j})] \\ -\frac{1}{2}\beta^n_{i,j}[x^n_{i+1,j+1} - x^n_{i+1,j-1} - x^n_{i-1,j+1} + x^n_{i-1,j-1}] \\ +\gamma^n_{i,j}[(x^n_{i,j+1} - 2x^n_{i,j} + x^n_{i,j-1}) + \frac{\psi_{i,j}}{2}(x^n_{i,j+1} - x^n_{i,j-1})] \end{aligned} \tag{10}$$

and

$$\begin{aligned} Ly^n_{i,j} = \alpha^n_{i,j}[(y^n_{i+1,j} - 2y^n_{i,j} + y^n_{i-1,j}) + \frac{\phi_{i,j}}{2}(y^n_{i+1,j} - y^n_{i-1,j})] \\ -\frac{1}{2}\beta^n_{i,j}[y^n_{i+1,j+1} - y^n_{i+1,j-1} - y^n_{i-1,j+1} + y^n_{i-1,j-1}] \\ +\gamma^n_{i,j}[(y^n_{i,j+1} - 2y^n_{i,j} + y^n_{i,j-1}) + \frac{\psi_{i,j}}{2}(y^n_{i,j+1} - yn_{i,j-1})] \end{aligned} \tag{11}$$

Standard central differences should be used to evaluate α, β and γ. The scheme is carried out in two sweeps:

- Sweep 1:

$$\begin{cases} (r - \alpha\delta_{\xi\xi})f^n_{x_{i,j}} = rwLx^n_{i,j} \\ (r - \alpha\delta_{\xi\xi})f^n_{y_{i,j}} = rwLy^n_{i,j} \end{cases} \tag{12}$$

- Sweep 2:

$$\begin{cases} (r - \gamma\delta_{\eta\eta})C^n_{x_{i,j}} = f^n_{x_{i,j}} \\ (r - \gamma\delta_{\eta\eta})C^n_{y_{i,j}} = f^n_{y_{i,j}} \end{cases} \tag{13}$$

The relaxation parameters are r and w and f_y and f_x are intermediary solutions. Every sweep represents a series of a tri-diagonal matrix inversions. Sweep 1 represents inversions in the ξ direction (along $\eta = const$). Sweep 2 represents inversions in the η direction (along $\xi = const$). For example, the equation for x of sweep 1 is set up and inverted as follows:

$$(r - \alpha\delta_{\xi\xi})f^n_{x_{i,j}} = rwLx^n_{i,j} \tag{14}$$

and the finite difference equation is:

$$\underbrace{-\alpha^n_{i,j}}_{A} f^n_{x_{i-1,j}} + \underbrace{(r + 2\alpha^n_{i,j})}_{B} f^n_{x_{i,j}} \underbrace{-\alpha^n_{i,j}}_{C} f^n_{x_{i+1,j}} = \underbrace{rwLx^n_{i,j}}_{D} \tag{15}$$

The boundary conditions for f_x, f_y, C_x, and C_y are zero (the boundary does not move), thus, for the above mentioned sweep direction:

$$\begin{aligned} A_0 = 0, \quad A_{i_{max}} = 0 \\ B_0 = 1, \quad B_{i_{max}} = 1 \\ C_0 = 0, \quad C_{i_{max}} = 0 \\ D_0 = 0, \quad D_{i_{max}} = 0 \end{aligned} \tag{16}$$

The tri-diagonal system that is solved looks as follows:

$$\begin{pmatrix} B_0 & 0 & 0 & \dots & \dots & 0 & 0 \\ A_1 & B_1 & C_1 & 0 & \dots & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & \dots & 0 \\ 0 & 0 & A_i & B_i & C_i & 0 & 0 \\ 0 & \dots & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & A_{i_{max}-1} & B_{i_{max}-1} & C_{i_{max}-1} \\ 0 & 0 & \dots & \dots & 0 & 0 & B_{i_{max}} \end{pmatrix} \begin{pmatrix} f_{x_0} \\ fx_1 \\ \dots \\ \dots \\ \dots \\ f_{x_{i_{max}-1}} \\ f_{x_{i_{max}}} \end{pmatrix} = \begin{pmatrix} 0 \\ D_1 \\ \dots \\ \dots \\ \dots \\ D_{i_{max}-1} \\ 0 \end{pmatrix} \quad (17)$$

The inversions of sweep 2 are conducted is a similar fashion, this time in the η direction. The boundary conditions for sweep 2 are the same as those of sweep 1.

The parameters r and w are used to accelerate the convergence and sometimes even prevent divergence. Usually one chooses $0 < \omega < 2$ and $r > 0$. A typical value for r is around 0.01 or 0.001. A typical value for ω is less then 2. For a coarse mesh one should take $\omega = 1.5$. In cases of high stretching values and/or a poor initial guess you should use larger values for r.

7 Results

Generate a mesh using the following input (indexes are for "C", indexes for Fortran are in parentheses[1]):

$$\begin{aligned} t &= 0.12 \\ i_{max} &= 50\ (51) \\ j_{max} &= 25\ (26) \\ i_{TEL} &= 11\ (12) \\ i_{LE} &= 25\ (26) \\ i_{TEU} &= 39\ (40) \\ \Delta y &= 0.02 \\ XSF &= 1.15 \\ YSF &= 1.15 \end{aligned}$$

Note that $ni = 51$ and $nj = 26$ for both "C" and Fortran.

[1]The origin of the computational coordinate system, (ξ, η) is $(i_{imin}, j_{min}) = (0, 0)$ for "C" while it is $(i_{imin}, j_{min}) = (1, 1)$ for Fortran

7.1 Cases

1. $\phi = \psi = 0$ without control.
2. $\phi = 0, \psi$ is calculated from the boundary conditions.
3. $\psi = 0, \phi$ is calculated from the boundary conditions.
4. ψ and ϕ are calculated from the boundary conditions.

Present your results using two graphs, a complete view and a close-up view of the airfoil. In addition, plot the convergence history. The solution is considered converged if the residual, defined by:

$$\begin{matrix} Log_{10}(|Lx_{ij}^n|_{max}) \\ Log_{10}(|Ly_{ij}^n|_{max}) \end{matrix} \tag{18}$$

has dropped 5 to 6 orders of magnitude. Use the program to analyze the influence of r, and w on the convergence.

A Management of an N-Dimensional Arrays as a 1-Dimensional Array

Let $A_{i_1,i_2,\ldots,i_n}$ be an "n" dimensional array, let the indexes run from inner most to outer most, *i.e.*, index i_1 is the first and i_n is the last, and let $n_1, n_2, \ldots, n_n$ be the size of the array, the index of a 1-D array, spanning the array A, would be:

$$\text{offsetnd} = (\ldots((i_n * n_{n-1} + i_{n-1}) * n_{n-2} + i_{n-2}) * n_{n-3} + \ldots) * n1 + i_1$$

For example, for a 2-D array, the index takes the form:

$$\text{offset2d} = j * ni + i$$

for a 3-D array, the index takes the form:

$$\text{offset3d} = (k * nj + j) * ni + i$$

and for a 4-D array, the index takes the form:

$$\text{offset4d} = ((l * nk + k) * nj + j) * ni + i$$

B Allocating and Accessing a 2-D Array

B.1 Allocation

```
double *a;
scanf("%d",ni, nj);
a = (double *)malloc((unsigned) ni * nj * sizeof(double));
```

B.2 Offset Function

```
int offset2d(int i, int j, int ni}
{
return j * ni + i;
}
```

B.3 Accessing an Array Element

Following the above array declaration and the definition of the offset2d function, accessing the i, j element of a 1-D array is conducted using:

```
a[offset2d(i, j, ni)]
```

B.4 Notes

1. You need only offset2d for the current exercise
2. Always (!) start your arrays from "0"
3. Note that for the current exercise $ni = 51$ and $nj = 26$

C General Structure of a CFD Program

The computer program must be written in a modular way and with lots of comments. The following subprograms should appear in an appropriate CFD program:

1. INPUT
2. OUTPUT
3. INITIALIZE - Calculations of coefficients.
4. JACOB, METRICS - Calculation of the metric coefficients (and the Jacobian where applicable).
5. RHS - Calculation of the source terms (RHS).
6. LHS - calculation of matrix elements.
7. SOLVE - A function that solves the linear system, *e.g.*, a function that implements the Thomas algorithm.
8. STEP - The heart of the program. It advances the solution by one time step. It invokes the calls to RHS, LHS, SOLVE, and finally updates the solution.

9. BC - Calculation of the boundary conditions.

10. MAIN - The main program. Usually invokes the calls to INPUT, INITIALIZE, and BC and performs a loop on STEP until convergence. The call to BC can be inside or outside the loop, based on the specific problem. The main program eventually invokes a call to OUTPUT.

D Additional Tips

You may use the same tridiagonal matrix solver function for this exercise as well. Note that the start and end arguments may change. Be aware that the solver changes the content of some of the input arrays.

Enjoy