Technion - Israel Institute of Technology
Faculty of Aerospace Engineering
Computational Aerodynamics
Exercise no. 2

Project Objective

Writing a general computer program to solve the Euler equations for a two dimensional flow in curvilinear coordinates using the Beam & Warming algorithm (using the Euler implicit, first order temporal accuracy, scheme)

The Computer Program:

1. Write a sub-program for input parameters.
2. Write a sub-program to read the grid.
3. Write a sub-program to calculate the metric coefficients.
4. Write a sub-program to calculate the boundary conditions of the following types:
   (a) An adiabatic wall for $j = j_{min}$, $i = i_{tel} \dots i_{teu}$ (tel = trailing edge lower; teu = trailing edge upper)
   (b) Cut type condition for $j = j_{min}$, $i = 1 \dots i_{tel}$, $j = j_{min}$ and $i = i_{teu} \dots imax$
   (c) Outlet conditions for $i = i_{min}$ and $i = i_{max}$.
5. Write a sub-program to evaluate the RHS.
6. Write a sub-program to evaluate the LHS.
7. Sub-programs for inversions, smoothing (RHS and LHS smoothing), and for the solution output can be found in the course home-page.
8. Combine the above mentioned sub-programs into a general program to solve the flow about the airfoil for which you created a computational mesh in project 1.

Governing Equations

The set of Euler equations in curvilinear coordinates are given by:

$$\frac{\partial \hat{Q}}{\partial \tau} + \frac{\partial \hat{E}}{\partial \xi} + \frac{\partial \hat{F}}{\partial \eta} = 0 \tag{1}$$

where

$$\hat{Q} = \frac{1}{J}\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix} \quad \hat{E} = \frac{1}{J}\begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ (e+p)\,U \end{bmatrix} \hat{F} = \frac{1}{J}\begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ (e+p)\,V \end{bmatrix} \tag{2}$$

where J is the Jacobian and U and V are the contravariant velocities that are given by the relation,

$$\begin{bmatrix} U \\ V \end{bmatrix} = \begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix} \tag{3}$$

Beam And Warming Algorithm

The first order Beam & Warming scheme including smoothing is given by:

$$\left(I + \frac{\triangle t}{2} D_{0_\xi}\hat{A}^n + \triangle t D_{is_\xi}\right)\left(I + \frac{\triangle t}{2} D_{0_\eta}\hat{B}^n + \triangle t D_{is_\eta}\right)\triangle \hat{Q}^n = RHS^n \tag{4}$$

Where the RHS^n is:

$$RHS^n = -\left(\frac{\triangle t}{2} D_{o_\xi}\hat{E}^n + \frac{\triangle t}{2} D_{0_\eta}\hat{F}^n\right) + D_{es_\xi}Q^n + D_{es_\eta}Q^n \tag{5}$$

Here, D_0 is a second order central difference operator:

$$D_{0_\xi} f_{ij} = f_{i+1,j} - f_{i-1,j} \tag{6}$$

The artificial dissipation terms D_{is} and D_{es} may be calculated using the sub-programs given in the course web-site (**smooth** for the explicit smoothing and **smoothx** and **smoothy** for the implicit smoothing).

The calculation of the RHS is being done in two steps:

1. Call the sub-program that calculates the RHS (should have been written by you).
2. Call the sub-program **smooth** (which is given in the course home-page).

For the LHS you will need the Jacobian matrices Â and B̂ that are given by:

$$\hat{A}, \hat{B} \quad = \quad \begin{bmatrix} k_t & k_x & k_y & 0 \\ k_x\phi^2 - u\theta & +\theta - k_x\gamma_2 u & k_y u - \gamma_1 k_x v & k_x\gamma_1 \\ k_y\phi^2 - v\theta & k_x v - k_y\gamma_1 u & k_t + \theta - k_y\gamma_2 v & k_y\gamma_1 \\ \theta(2\phi^2 - \frac{\gamma e}{\rho}) & k_x\beta - \gamma_1 u\theta & k_y\beta - \gamma_1 v\theta & \gamma\theta \end{bmatrix} \tag{7}$$

where

$$\begin{aligned}
\phi^2 &= 0.5(\gamma-1)(u^2+v^2) \\
\theta &= k_x u + k_y v \\
\gamma_1 &= \gamma - 1 \\
\gamma_2 &= \gamma - 2 \\
\beta &= \frac{\gamma e}{\rho} - \phi^2
\end{aligned}$$

where $k = \xi$ to obtain $\hat{A}$ and $k = \eta$ to obtain $\hat{B}$.

Calculating the two sweeps of the LHS is being done by two sub-programs (for each sweep a separate sub-program) and by calling the sub-programs **smoothx** and **smoothy**. One for calculating $\triangle t D_{is_\xi}$ and the other for calculating $\triangle t D_{is_\eta}$. Inversion of the tri-diagonal block system is being done using the sub-program **btri4s**.

The Sub-program Step

The sub-program **step** should be written as follows:

1. Zero the RHS array.

2. Calculate the RHS.

3. Add smoothing to the RHS array.

4. Solve the system in two steps:

   (a) <u>First step, ξ-sweep:</u>

   - Calculate $\frac{\triangle t}{2} D_{0\xi} A^n_{ij} + \triangle t D_{is_\xi}$ for a specific j ,and place it in a work array.
   - Place the corresponding row of RHS in another work array.
   - Solve the system using **btri4s**.
   - Place the results back in the RHS array.

   (b) <u>Second step, η-sweep:</u>

   - Similar to the first step, this time by columns.

   (c) Advance the solution with

$$Q^{n+1}_{ij} = Q^n_{,ij} + S^n_{ij} * J_{ij} \tag{8}$$

The **step** sub-program is called from the **main** and should be looped upon until convergence.

Boundary Conditions:

Write a separate sub-program for the boundary conditions which calls the sub-program for calculating the specific boundary conditions as required. The call to that sub-program is being made from the **main** right after the call to **step**.

Appendix 1 - Debugging Stages

- Step A, Checking the RHS:
  1. Start with a simple Cartesian mesh
  2. Calculate the RHS with Eq. 5 and advance the program with Eq. 8 without imposing the boundary conditions !!! In this step, even with a relatively large time step, Q does not change.
  3. Reduce the time step and try it on the "real" mesh. The flow field changes slowly due to mesh imperfections, round off errors, and metric calculations.
  4. Enable the call to the boundary conditions (it is now necessary to chose a very small time step that is appropriate for an explicit scheme), run a large number of steps for a Mach number of $M_\infty = 0.3 - 0.4$.
- Step B, Checking the LHS:
  1. Disable the boundary conditions (start with a simple mesh) and run for a large time step. On a Cartesian mesh, Q should not change.
  2. Repeat that with a small time step.
  3. Enable the boundary conditions and rerun.

You have been warned, **IT IS NOT ADVISED** to attempt to write and debug your program without following the above recommended steps

Appendix 2 - Input for the Provided Sub-Programs

- **smooth**: Except for Q,S and the metric coefficients, it is necessary to pass four, one-dimensional work arrays in the size MAX(ID,JD). It is necessary to pass the Mach number in infinity - FSMACH, $\gamma(= 1.4)$, EPSE - smoothing coefficient (around 0.06) and $\triangle t$.

- **btri4s:** MD = MAX(ID,JD) (only in Fortran). KS, and KE should be set to 1 and ID - 2 or JD - 2 (depending on the inversion direction, they should be set to 2 and ID - 1 or JD - 1 in Fortran)

- **smoothx, smoothy:** Similar to Smooth with the addition of I or J. it is also necessary to provide the tri-diagonal A,B,C blocks.

Appendix 3 - Results and Report

1. Calculate the flow field about the airfoil at $M_\infty = 0.9$ and $M_\infty = 1.5$. Plot the convergence history, and Mach and pressure distributions on the airfoil. Use $L_2 Norm$ for the convergence criterion.
2. Study the effect of at least one of the following on the results (or even a combination of two of the following items).
   (a) Time step.
   (b) Smoothing coefficient.
   (c) Outlet boundary condition.
   (d) Increased grid resolution.

Good Luck !!!