Project 1: Flow over a Hanger
AE523, Computanial Fluid Dynamics, Fall 2016

Tzu-Hsiang Lin *

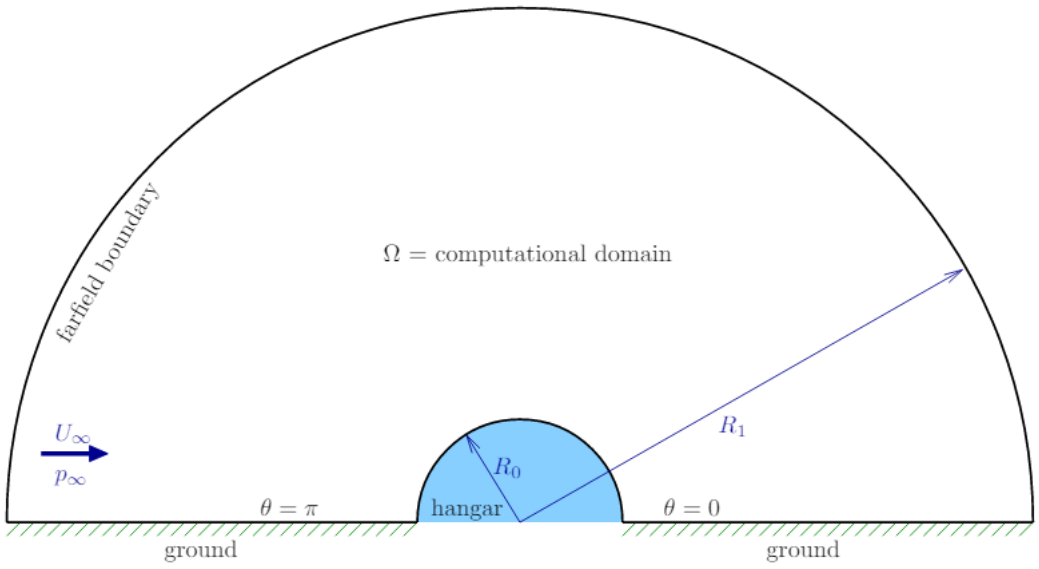
Abstract

Consider a simple fluid dynamics case: non-viscous, incompressible and steady flow over a half-cylinder. In this report, different numerical approaches will be applied for mapping and grid establish, as well as the different schemes, directly solver by sparse matrix and iterative solver by multi-grid, will be used for solving the partial derivative equations. Numerical results will be turned into different coefficients, C_l C_p, to compare the accuracy and calculating efficiency.

KEYWORDS: Sparse matrix, Multigrid , Finite difference method

1 Case Description

Considering a half-cylinder building of radius R_0. Of interest is the flow outside the hangar, and so the computational domain consists of the air outside the cylinder, in a semi-circular region of $R_1 = 20R_0$. Figure 1 shows the setup for this problem. The uniform flow from the left far way side is $U_\infty \hat{x}$. As the air flow over the half-cylinder, the velocity changee, so does the pressure, causing a force on the cylinder. Assuming the pressure inside the cylinder is P_{inf}, same as the static pressure of the free stream flow.



[1]tzuhslin@umich.edu

fig. 1: Setup for the calculation of flow over a half-cylinder

Fluid Model

Assuming the fluid is incompressible, $\nabla \cdot \vec{v} = 0$, and irrotational, $\nabla \times \vec{v} = 0$, where $\vec{v} = u\hat{x} + v\hat{y}$ is the velocity vector. Define a stream function, $\psi(\hat{x})$, where $\hat{x} = x\hat{x} + y\hat{y}$, therefore, the velocity components are

$$u = \frac{\partial \psi}{\partial y}, v = -\frac{\partial \psi}{\partial x} \tag{1}$$

Since it is an irrotaional flow, then the governing equation will be

$$\nabla^2 \psi = 0 \tag{2}$$

Once the governing equation has been define, the relative coefficients can also be achieved by relative equations. Here, the pressure distribution can be found from Bernoulli's equation,

$$p(\vec{x}) + \frac{1}{2}\rho|\vec{v}(\vec{x})|^2 = p_\infty + \frac{1}{2}\rho U_\infty^2 \tag{3}$$

Integrating the pressure over the surface of the cylinder gives the 2D forces on the cylinder,

$$\vec{F'} = \int_{sufeace} (p - p_\infty)\vec{n}dl = D^{'}\hat{x} + L^{'}\hat{y} \tag{4}$$

The lifting forces obtained from the Equation 4 can be non-dimensionalized into lift coefficients, as well as the pressure from Equation 3.

$$C_l = \frac{L^{'}}{\frac{1}{2}\rho U_\infty^2 2R_0}, C_p = \frac{p - p_\infty}{\frac{1}{2}\rho U_\infty^2} \tag{5}$$

Numerical Approach

Before starting, some initial air parameters will be defined. After that, three main steps will be applied to solve the case. First, establish the grids and transfer to the reference computational spaces. Second, two different numerical schemes, sparse matrix and multigrid, will be applied to solve the governing equation and find the stream functions. Third, after obtaining the stream functions in the global domain, post processes will take place and find out the lift and pressure coefficient.

Initial parameters set up

In order to simplify the problem, set the air density, pressure, velocity of the uniform flow and the radius of the half-cylinder as 1. Therefore, $\rho = 1, U_\infty = 1, p_\infty = 1, R_0 = 1$.

Establish the Grids

There are two different ways to establish the computational grids in the global domain. First is to separate the boundary uniformly, as shown in the figure 2(a), for N=8, and 2(b), for N=16. This is the easiest way to make the computational grids. However, we only care about the flow along or near the cylinder boundary, which will directly influence the pressure distribution,

(a) N=8 (b) N=16

fig. 2: The uniform computational grid in global domain.

(a) N=8 (b) N=16

fig. 3: The compressed computational grid in global domain.

therefore, we could decrease the density of the grids far away from the cylinder boundary to save some computational resources. So, the second method is to separate the grids with a certain space which will increase along with the distance between itself and the boundary. As shown in figure 3(a), for N=8, and figure 3(b), for N=16.

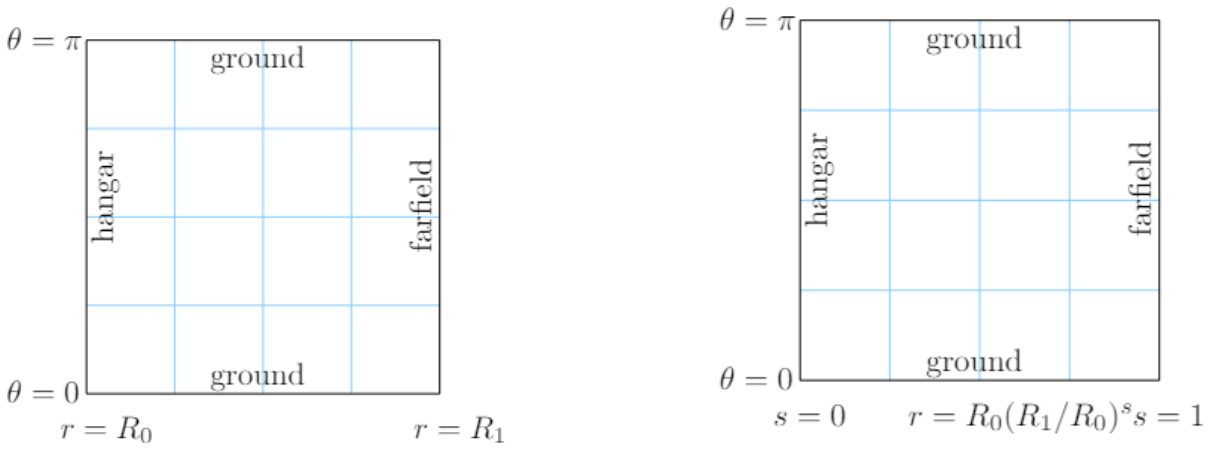
Transfer to reference computational domain

Since the spaces of the grid in global domain are not rectangular either in uniform grid or compressed grid. To simplify the computational processes, we transfer the grid in global domain into the reference domain. In the reference computational domain, for both uniform grid and compressed grid are rectangular, as the figure 4.

Numerical Solver

Direct solver by using Sparse matrix

After having the computational domain, we transfer our governing equation in terms of the coordinate in the reference domain. Therefore, the Equation 2 can be written as:



(a) Uniform grid

(b) Compressed grid

fig. 4: Uniform and compressed grid in computational reference domain.

Table 1: The value of the stream function on uniform separated and compressed grid.

(a) The value of the stream function on uniform separated grid with N=3

0	0	0	0
0	6.3836	11.9491	17.3205
0	6.3836	11.9491	17.3205
0	0	0	0

(b) The value of the stream function on compressed grid with N=3

0	0	0	0
0	6.3836	11.9491	17.3205
0	6.3836	11.9491	17.3205
0	0	0	0

$$\nabla^2\psi = \frac{\partial^2\psi}{\partial r^2} + \frac{1}{r^2}\frac{\partial^2\psi}{\partial\theta^2} + \frac{1}{r}\frac{\partial\psi}{\partial r} = 0 \tag{6}$$

Now, apply the finite difference method to break down the Equation 6, then we can have the following form of the Equation 6.

$$\frac{\psi^n_{i+1,j} - 2\psi^n_{i,j} + \psi^n_{i-1,j}}{\Delta r^2} + \frac{\psi^n_{i,j+1} - 2\psi^n_{i,j} + \psi^n_{i,j-1}}{\Delta\theta^2} - \frac{1}{r}\frac{\psi^n_{i,j+1} - \psi^n_{i,j-1}}{2\Delta r} = 0 \tag{7}$$

Then, gathering the center node, $\psi^n_{i,j}$, the left, $\psi^n_{i-1,j}$, the right,$\psi^n_{i+1,j}$, the up $\psi^n_{i,j+1}$, and the down, $\psi^n_{i,j-1}$. So, the Equation 7 can be transfer into the matrix form, and the coefficient matrix of the corresponding point is named as matrix A. The corresponding codes are listed in the Appendix A.

$$\underline{\underline{A}}\,\underline{\psi} = \underline{Q} \tag{8}$$

Here, the source term, Q matrix, represent as the boundary condition. Once the A matrix and Q matrix are obtained, the ψ matrix can be solved. Table 1 shows the result of ψ when N=3 on different grid, here, N is the number of spaces that separate the ground on the each side of the half-cylinder.

Iterative solver by using Multi-grid

An alternative method to solve the the Finite difference in Equation 7 is by using iterative method. For solving the Boundary Condition Problem, Multi-grid method is quite efficiency. Though there are several forms of Multi-grid scheme, only the V cycle scheme will be discussed in this project. Still base on the reference computational domain of the compressed grid. Some parameters and initial conditions for the multi-grid method is listed below.

- N as a number of power of 2
- N=4 for coarsest mesh
- Red-black Gauss-Seidel smoothing
- $\nu_1 = \nu_2 = 2$ for pre/post-smoothing iterations
- $\nu_c = 10$ smoothing iterations for the coarse solve
- Full-weighting residual restriction
- Prolongation via interpolation
- Free-stream initial condition of $\psi = U_\infty y$

Once have the initial stream function from free stream, applied two times Gauss-Seidel smoothing to the initial ψ matrix then get ψ^*. Then, substitute the ψ^* matrix back to the Equation 7. If the equation does not satisfy, then the difference between the source term and the answer from ψ^* is the residual. Later, restrict the residual down to the next level with a coarser mesh, and initialize the stream function by setting ψ as 0 and repeat the previous steps till reach the coarsest mesh; then, prolongate back to the first level and a V-cycle is completed. Repeat the whole V-cycle till the residual converges to a desired value. The figure 5 indicates how the L1 norm of the residual converges with the number of V-cycle and the working units. And the working unit can be defined as:

$$1V - Cycle = \sum_{l=0}^{n_level-1} (\nu_1 + \nu_2)^{-l} workunits \tag{9}$$

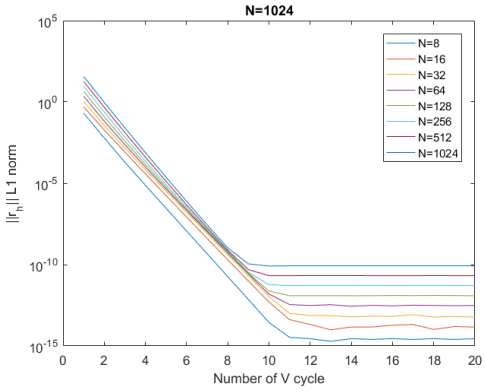
For N equal to 256 or less, the working units are too small, so will not be seen in the figure 5(b). In table 2 and figure 6, the results from two different solver with N=16 are compared. Also the corresponding codes are in the Appendix B.

Post-processing

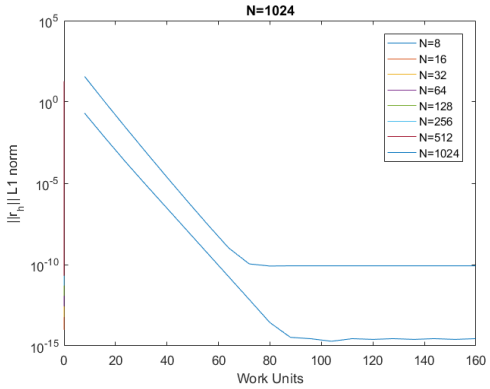
As soon as obtaining the stream function on each node of the grid, we can plot out the streamlines(contour lines of ψ) in the global space and pressure coefficient on the half-cylinder. Here, the coefficient of lift will also be discussed.

Streamline contour

Setting N=8 and N=32 to compare the streamline in the global domain, with different grid. Figure 7 shows the streamlines and the stream function contour for both N=8 and N=32 with uniform spaced grid. Figure 8 shows the streamlines and the stream function contour for both N=8 and N=32 with compressed grid.

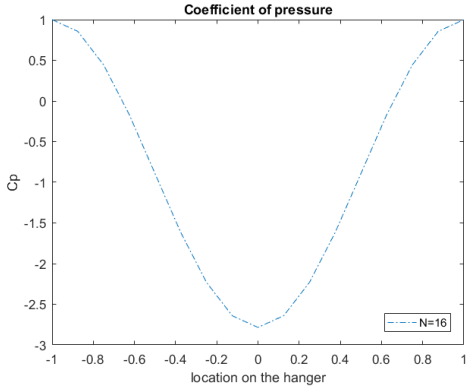


(a) L1 norm of the residual converges with the number of V-cycle

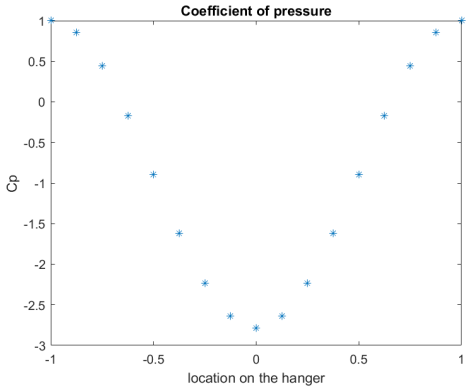


(b) L1 norm of the residual converges with the working units

fig. 5: L1 norm of the residual converges with the number of V-cycle and the working units



(a) streamline in uniform grid global domain with N=8.



(b) streamline in uniform grid global domain with N=32

fig. 6: The streamline in uniform grid global domain with N=8 and N=32

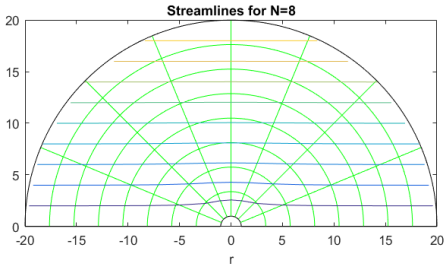
Table 2: The lift coefficient for different N number in both Sparse Matrix solver and Multigrid solver.

(a) Direct solver with Sparse Matrix

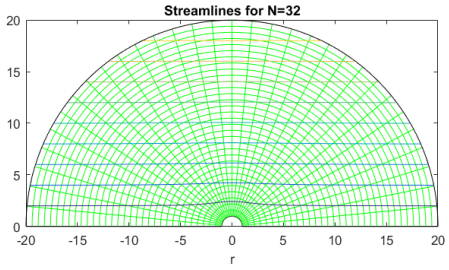
N	C_l	computational time
8	1.2592	0.4375 s
16	1.5349	0.5000 s
32	1.6370	0.7188 s
64	1.6683	0.875 s
128	1.6770	2.2656 S

(b) Iterative solver with Multi-grid V-Cycle

N	C_l	computational time
8	1.2615	0.5625 s
16	1.5396	0.5469 s
32	1.6385	0.5938 s
64	1.6687	0.7969 s
128	1.6771	1.2969 S

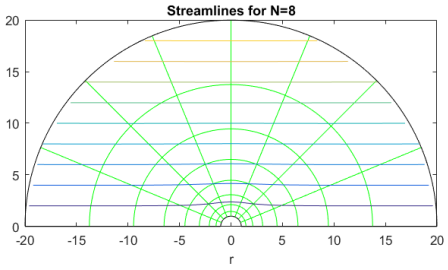


(a) streamline in uniform grid global domain with N=8.

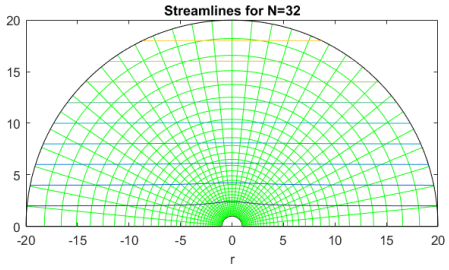


(b) streamline in uniform grid global domain with N=32

fig. 7: The streamline in uniform grid global domain with N=8 and N=32

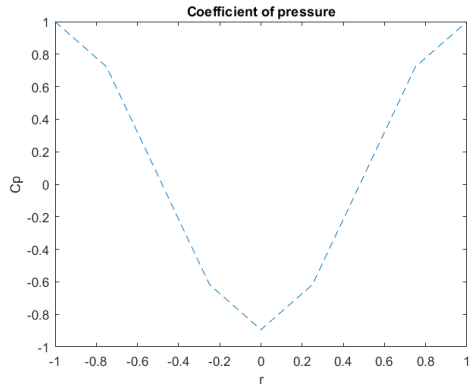


(a) streamline in compressed grid global domain with N=8

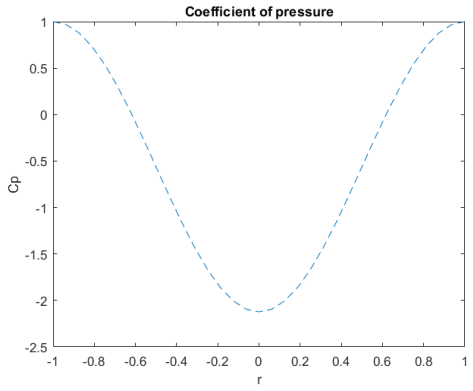


(b) streamline in compressed grid global domain with N=32

fig. 8: streamline in compressed grid global domain with N=8 and N=32

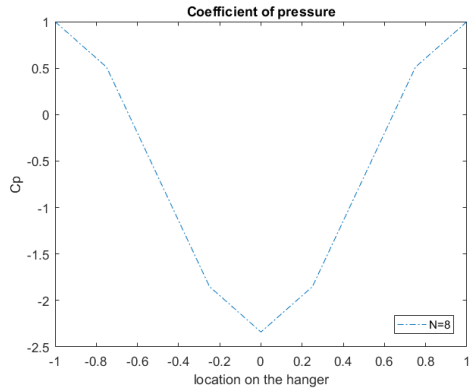


(a) Pressure coefficient in uniform grid global domain with N=8

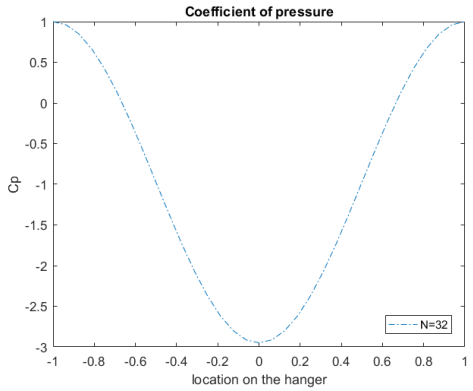


(b) Pressure coefficient in uniform grid global domain with N=32

fig. 9: Pressure coefficient in uniform grid global domain with N=8 and N=32



(a) Pressure coefficient in compressed grid global domain with N=8



(b) Pressure coefficient in compressed grid global domain with N=32

fig. 10: Pressure coefficient in compressed grid global domain with N=8 and N=32

Pressure coefficient

For pressure coefficient, also set N=8 and N=32 to compare the pressure coefficient along with the surface of the half-cylinder. Figure 9 shows the pressure coefficient for both N=8 and N=32 with uniform spaced grid. Figure 10 shows the pressure coefficient for both N=8 and N=32 with compressed grid.

Lift coefficient

Integrating the pressure distribution along the surface of the half-cylinder can gain the force. Since the lifting force is perpendicular to the free stream, only the y-direction force will be considered. Table 3 shows the 2D lift coefficient for both N=8 and N=32 with uniform spaced grid. It is clear to see that the lift coefficient for both uniform grid and compressed grid are not accuracy when N equal to a small number. However, for compressed grid, the error between approximate C_l is much smaller. Similar trend can also be seen in figure 11. The order of the error for C_l in compressed grid is under than 1 percent, however, for uniform grid is just near 10 percent. Also the computational time for each grid are quit similar.

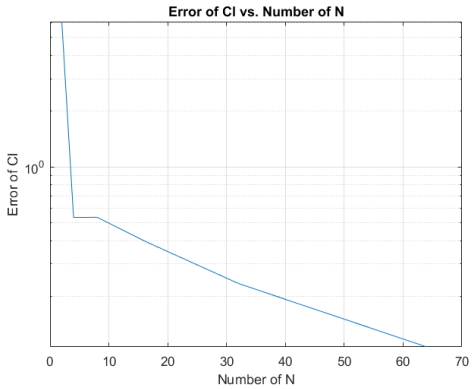
Table 3: The lift coefficient and computational time for different N number in both uniform grid and compressed grid.

(a) Uniform grid

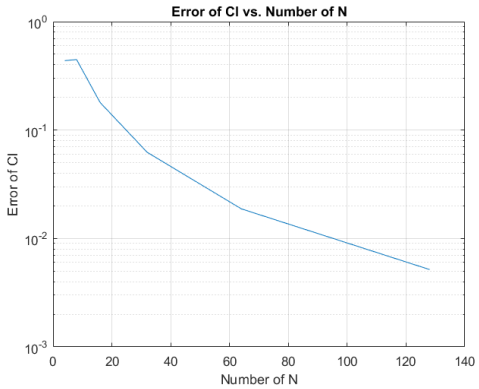
N	C_l	computational time
8	0.3031	0.3438 s
16	0.6524	0.4219 s
32	1.0859	0.5000 s
64	1.4194	0.7188 s
128	1.5891	2.2344 s
256	1.6528	36.1875 s
512	1.6725	732.5000 s

(b) Compressed grid

N	C_l	computational time
8	1.2592	0.4375 s
16	1.5349	0.5000 s
32	1.6370	0.7188 s
64	1.6683	0.875 s
128	1.6770	2.2656 s
256	1.6793	35.7813 s
512	1.6799	696.5000 s



(a) Uniform grid



(b) Compressed grid

fig. 11: Error of the coefficient of lift versus the number of N in uniform grid and compressed grid.

Using a large N to find the "Exact" solution

A solution very close to the exact solution can be obtained by applying a large N number. Here, setting N=512, and the value of coefficient of lift is 1.6799, which is very close to the exact solution 1.677.[2] Figure 12 shows how the error in coefficient of lift converges with N.

Appendix

1. A matrix for uniform grid

```
%==========Build up the matrices==========%

for j=1:N+1

    for i=2:N
        if j>=2 && j<=N
                m=(j-1)*(N+1)+i;
                r=R0+(i-1)*dr;
```

```
            % A matrix
            A(m,m)=-2*(dr^-2+(r*dtheta)^-2);          % Note:
            A(m,m+1)=dr^-2 + (2*r*dr)^-1; % East side    R0+i*dr = r
            A(m,m-1)=dr^-2 - (2*r*dr)^-1; % West side
        A(m,m+(N+1))=(r*dtheta)^-2;  % North side
            A(m,m-(N+1))=(r*dtheta)^-2;  % South side

            % Source term matrix
            q(m)=0;
        end
    end

    %==================Boundary Conditions============================%
        %hanger
        m=(j-1)*(N+1)+1;q(m,1)=0;
        %farfield
        m=(j-1)*(N+1)+N+1;q(m,1)=R1*sin((j-1)*dtheta);
        %grounds
        q(j,1)=0;m=N*(N+1)+j;q(m,1)=0;
end
```

2. A matrix for compressed grid

```
or j=1:N+1

    s=(j-1)*ds;

    %================ Filled up the A matrix ======================%
    if j>=2 && j<=N
        for i=2:N
            m=(j-1)*(N+1)+i;
            C1 = (log(R1/R0)*(R1/R0)^s)^-2;
            C2 = (R1/R0)^-(2*s);
            % A matrix
            A(m,m)=-2*(C1*ds^-2 + C2*dtheta^-2);
            A(m,m+1)= C1*ds^-2;              % East side
            A(m,m-1)= C1*ds^-2;              % West side
            A(m,m+(N+1))= C2*dtheta^-2;  % North side
            A(m,m-(N+1))= C2*dtheta^-2;  % South side
            % Source term matrix
            q(m)=0;
        end
    end

    %================ Boundary Conditions ========================%
    %hanger
```

```
    m=(j-1)*(N+1)+1;q(m,1)=0;
    %farfield
    m=(j-1)*(N+1)+N+1;q(m,1)=R1*sin((j-1)*dtheta);
    %grounds
    q(j,1)=0;m=N*(N+1)+j;q(m,1)=0;

end
```

3.Multigrid method scheme

```
N=2^n; V=0;
    % ================== Restriction ================================ %
    while 1
        V=V+1; L=0;
        while N~=4
            % Parameters
            if L~=0; N=N/2;end
            L = L+1; %N= 2^n; %L:Level; N:Grid point
            d_s = 1/N; dtheta = (theta1-theta0)/N;
            C1 = (log(20)*d_s)^-2; C2 = dtheta^-2;

            % Initialize the Psi matrix
            if L==1 && V==1
                Psi_h{L} = build_up_psi(N,d_s,dtheta,R0,R1,U_inf);
                fh = zeros(N+1,N+1); f{1} = fh;
            elseif L == 1
                Psi_h{L} =  Psi{V-1};
            else
                Psi_h{L} = zeros(N+1,N+1);
            end

            % smooth the Psi
            Psi_s{L} = smoothing(Psi_h{L},C1,C2,N,f{L});

            % Calculate the residual
            r_h{L} = residual(Psi_s{L},C1,C2,N,f{L});

            % Restrcit to the next level
            fH = restriction(r_h{L},N);
            f{L+1} = fH;
        end

    % ================ Prolongation ================================ %

        while L>=2
            N = N*2;
            d_s = 1/N; dtheta = (theta1-theta0)/N;
```

```
        C1 = (log(20)*d_s)^-2; C2 = dtheta^-2;
        Psi_s{L-1} = Psi_s{L-1} + prolongation(Psi_s{L},N/2);
        Psi_s{L-1} = smoothing(Psi_s{L-1},C1,C2,N,f{L-1});
        L = L-1;
    end

% ========== Residual for a completed V cycle ========== %

    Psi{V} = Psi_s{L}; % store the updated Psi_s to the nth V cycle
    tol = 1E-18*ones(N+1,N+1);
    normL1(V) = sum(sum(abs(r_h{L})))/(N+1)^2;

% ========== Define the Work Unit ========== %

    WU = 0; L = log2(N);
    for nlevl = 1:L
        WU = WU + 4*2^(-(nlevl-1));
    end
    WU(n-2)=WU;
```

Reference

[1] John Anderson: *Computational Fluid Dynamics: The Basics With Applications*, McGraw-Hill 1995.

[2] John Anderson: *Fundamental of Aerodynamics*, McGraw-Hill 2007.

[3] Hirsch,Charles: *Numerical Computation of Internal and External Flows,* vol.I:*Fundamentals of Numerical Discretization,* Wiley, New York, 1988.