**Project 1 for MEN 614 CFD Course**

**Grid Generation Using Elliptic PDEs for Cylinder**

**Wei, Tian**

**C#11345638**

Contents

[1 Introduction 1](#_Toc382898231)

[2 Governing Equations 2](#_Toc382898232)

[3 Discretization Schemes 2](#_Toc382898233)

[4 Boundary Condition Treatment 4](#_Toc382898234)

[5 Codes Structure 5](#_Toc382898235)

[6 Results and Discussion 6](#_Toc382898236)

[Appendix-codes 9](#_Toc382898237)

# Introduction

Computational Fluid Dynamics, abbreviated as CFD, is all about how to solve numerically Navier Stokes equations to approach the true solution. Many factors can affect the quality and accuracy of CFD, such as the numerical schemes, temporal and spatial steps, or how the space is discretized. Usually algebraic method and elliptic method are developed for grid generation.

Algebraic method is straightforward and easy to be implemented. After determine the start point and end point of a line, the line can be discretized linearly or exponentially. By linear methods points will evenly distributed along the line while by exponent method more point clusters to one side of the line. A much finer grid near the surface is significantly important to understand the flow in the boundary layer whose thickness is small.

Algebraic methods suits for most of the space discretization. One can arbitrarily determine the start point and end point on the edge of the space as desired. However, there is one drawback of this method when confronting irregular shape. After generating the grids, it is difficult to do the equation discretization since there would me thousands of different grid shapes. Different shapes mean that one has to discretize the governing equation into different forms. This is going to be very challenging especially for large space. If one simplified the equation discretization the accuracy of the solution can be deteriorated.

To solve this challenge the elliptic methods, or differential equation method, can be used. Elliptic method is to generate the grids by solving the Laplace equation that associate the physical domain with a computational domain. The physical domain is the actual space while computational space is a uniform grid that is used to discretize the equation. With a uniform girds of computational domain the equation discretization can be easy regardless of how complex the physical domain is.

The differential equation is solved numerically after specifying the boundary conditions. User can decide how many points and how they are distributed on the boundary. This process is very similar with that in the algebraic method. Then the discretized equation will be applied to all the cells in the computational domain and further be solved by numerical scheme, such as, Gauss-Seidel Point sweep method, Gauss-Seidel Line sweep method, or Gauss-Seidel Line sweep with a SOR method. The generated mesh of physical domain will be used to reconstruct the governing equations in order to make them applied to the computational domain.

This paper reports the efforts to implement the entire process described above. The quality of the grids by different boundary types and numerical schemes is discussed. Besides, the performance of numerical scheme is discussed by studying the relation of residual drop with the iteration time steps and time cost for convergence. Results shows that Gauss-Seidel Line sweep with SOR of 1.5 can achieve fast convergence speed and the quality of created mesh is much consistent with that from other numerical schemes.

# Governing Equations

The governing equations are shown as Equ.1 to Equ.5. Two variable *x* and *y* is the coordinates of the point in the mesh of physical domain. The point is corresponding to the pint in the computational domain uniquely. The subscripts represent two dimensions in the computational domain.

It is necessary to note that the coefficients which are the function of derivative of x and y is excluded from the second order derivative term because we want to have a linear discrete equation system so that the Gauss-Seidel series linear equation solver can be used. This may results in stability problem when solving the equation. Apparently the nonlinear equation solver can be used when the coefficients are merged in to second order derivative terms.

|  |  |
| --- | --- |
|  | 1 |
|  | 2 |
|  | 3 |
|  | 4 |
|  | 5 |

# Discretization Schemes

The above 5 equations can be discretized as the following equations. The first order derivative is discretized using central differencing. The second order derivative is discretized using three points.

|  |  |
| --- | --- |
|  | 6 |
|  | 7 |
|  | 8 |
|  | 9 |
|  | 10 |
|  | 11 |
|  | 12 |
|  | 13 |
|  | 14 |
|  | 15 |

Substitute Equ.6 to Equ.15 to Equ.1 to Equ.5, we can have the Equ.16 and Equ.17 for Gauss-Seidel Point Solver. Note that coefficients are calculated using the most-updated information. Equ.18 and Equ.19 is for Gauss-Seidel Line sweep and Gauss-Seidel Line sweep with SOR given that the sweep is from in axis from left to right.

When using Gauss-Seidel Line sweep, Tridiagonal Matrix Algorithm, abbreviated as TDMA, is used for solving the equations for the k+1 step after the equations for line=i are established. After line i is finished then go to line i+1. When solving equations for line i+1, the updated information of line i will be used immediately.

Equ.20 and Equ.21 show how to use Gauss-Seidel Line sweep with SOR method. SOR is a coefficient which in this paper is set to be larger than one in order to speed up the computation. This method is called over-relaxing. After implementing the Gauss-Seidel Line sweep for line i, the information of on that line will be updated as Equ.20 and Equ.21. The updated value is larger than that computed by the Gauss-Seidel Line Sweep method.

|  |  |
| --- | --- |
|  | 16 |
|  | 17 |
|  | 18 |
|  | 19 |
|  | 20 |
|  | 21 |

Residual of every point can be obtained by removing the RHS of Equ.16 and Equ.17 to the LHS. Residual needs to be check after finishing iteration once. Iteration needs continuing until the maximum of the residual is less than machine zero point which is 1e-12.

# Boundary Condition Treatment

There are four boundaries specified which are on the inner circle, outer circle which is located at 5 radius of the inner circle, and the horizontal lines connecting inner circle and outer circle, shown as Figure 1.

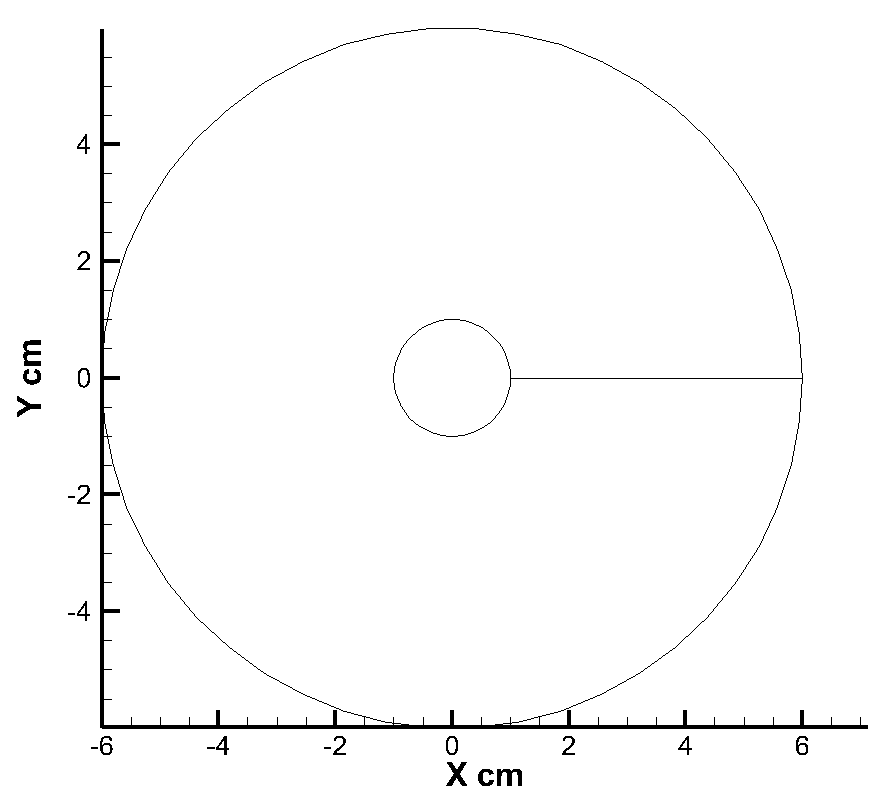


Figure boundary conditions

On the circle points are evenly distributed. Given that there are n points on each circle and points start from two points of the horizontal line in Figure1 in clockwise direction, we can have the coordinates of n points on outer circle using the Equ.22 and Equ.23, and points on the inner circle using Equ.24 and Equ.25. Note that i is from 1 to n.

|  |  |
| --- | --- |
|  | 22 |
|  | 23 |
|  | 24 |
|  | 25 |

On the horizontal line Equ. 26 is used to generate evenly distributed points and inner-circle-clustering points by adjust value of a. When a is equal to 1, uniform points are generated. When a is larger than 1, stretched points are generated. Equ.27 is the y coordinates of the points which entirely are zero. Note that j is from 0 to n

|  |  |
| --- | --- |
|  | 26 |
|  | 27 |

The entire computational domain can be denoted as the points from (1, 1) to (imax+1, jmax+1), given that we have *imax* increments in X and *jmax* increments in Y. Then we already know the values on the four boundaries which can be expressed as (1, j), (imax+1, j), (i, 1), and (i, jmax+1).

For the Gauss-Seidel Point method no boundary treatment is needed. Apply the Equ.16 and Equ.17 to the points from (2,2) to (imax, jmax) one iteration can be finished.

For the Gauss-Seidel Line sweep method and Gauss-Seidel Line sweep with SOR the boundary treatment is needed at start point (i,2) and end point (i, jmax) of each vertical line since one term on the LHS of Equ.18 and Equ.19 is known. For the start point, has to been more to RHS while for the end point has to be moved to the RHS. Then TDMA can be used to solve the Equ.18 and Equ.19 to get the updated information.

# Codes Structure

Codes structure is shown in Figure2:

* rs.f----subroutine to check the residual of all points in computational domain after one iteration
* tdma.f----subroutine to provide Tridiagonal Matrix Algorithm
* gsp.f---- Gauss-Seidel Point methods which calls rs.f to control iteration
* gsl.f---- Gauss-Seidel Line Sweep methods which calls rs.f to control iteration and tdma.f for computation line by line
* gsl\_sor.f---- Gauss-Seidel Line Sweep with SOR methods which calls rs.f to control iteration and tdma.f for computation line by line
* grid.f----main routine for control the computation which directly calls gsp.f, gsl.f and gsl\_sor.f

Figure 2 Structure of the codes

grid.f

gsp.f

gsl.f

gsl\_sor.f

rs.f

tdma.f

rs.f

# Results and Discussion

The results are obtained under the following settings:

* imax=30
* jmax=30
* a=enumeration (1,2); 1 is for uniform and 2 is for stretched points on horizontal line
* SOR=1.5; this value is obtained through multiple tests. SOR larger than 1.5 will crush down the computation. SOR=1.5 is the minimal number that guarantees stability of computation.

Figure 3 and Figure 4 show the uniform grids and stretched grids generated by Gauss-Seidel Point method, Gauss-Seidel Line sweep method and Gauss-Seidel Line sweep with SOR method. Results show that three methods can generate the almost same meshes.

|  |  |
| --- | --- |
| By Gauss-Seidel Point Methods | By Gauss-Seidel Line Sweep Methods |
| By Gauss-Seidel Line Sweep with SOR Methods | |
| Figure uniform grids generated by three methods | |

|  |  |
| --- | --- |
| By Gauss-Seidel Point Methods | By Gauss-Seidel Line Sweep Methods |
| By Gauss-Seidel Line Sweep with SOR Methods | |
| Figure stretched grids generated by three methods | |

Figure 5 show the relationship of residual drop in X and Y with iteration times. The iteration times for uniform grids and stretched grids generation are almost same. The residual drops with iteration times for all the methods, among which Gauss-Seidel Line Sweep with SOR (1.5) has the fastest convergence speed while Gauss-Seidel Point has the slowest convergence speed. Table1 shows the detailed data.

|  |  |
| --- | --- |
|  |  |
| Figure stretched grids generated by three methods | |

Table iteration steps of each method

|  |  |
| --- | --- |
| **Method** | **Iteration Steps** |
| Gauss-Seidel Point | 2332 |
| Gauss-Seidel Line Sweep | 170 |
| Gauss-Seidel Line Sweep with SOR of 1.5 | 68 |

# Appendix-codes