Computational Fluid Dynamics

(AE 706)

A report on

Assignment-2 Transfinite Interpolation

Instructor:

Prof. J C Mandal

Submitted By:

Ram Naresh Mahato

Roll No: 24M0065

Department of Aerospace Engineering

Indian Institute of Technology Bombay

Date: March 5, 2025

Table of Contents

1.Derivation

The physical domain is in Cartesian (x, y) co-ordinate while the computational domain is in (ξ, η) co-ordinate. The physical domain is bounded by (a) An ellipse of semi-major axis a=1 m and semi-minor axis of b= 0.5 m and (b) far-field boundary is circle with radius of R=20m. The domain is simply connected after introducing a cut, and it can be mapped to a unit square in computational domain (ξ, η) where, $\xi \in [0,1]$ and $\eta \in [0,1]$.

1.1 Boundary Conditions

a. The bottom boundary: The bottom boundary is an ellipse where η=0.

b. The top boundary: The top boundary is a circle and acts as a far-field condition where η=1.

c. The left boundary: This boundary represents the cut connecting the top and bottom boundaries where ξ=0.

d. The left boundary: This boundary represents the cut connecting the top and bottom boundaries where ξ=1.

1.2 Boundary in Physical domain in their Parameterization

a. The bottom boundary: Here, centre is at $(x_c, y_c) = (0, 0)$, a= semi-major axis and b= semi-minor axis. The parametric equation for ellipse is given as: $x(\theta)= x_c+ a\cos(\theta)$, $y= y_c + b\sin(\theta)$

where $\theta \in [0,2\pi)$. Mapping the parametric angle θ to the normalized parameter $\xi \in [0,1]$.

Hence, $\theta= 2\pi\xi$. The above equations reduce to: $x(\xi)= a\cos(2\pi\xi)$, $y(\xi)= b\sin(2\pi\xi)$.

This can finally be represented as $F(\xi,0) =[x(\xi,0), y(\xi,0)] = [\cos(2\pi\xi), 0.5\sin(2\pi\xi)]$.

b. The top boundary: The top boundary is a circle having cantered at $(x_c, y_c) = (0, 0)$ and radius R= 20m. The parametric equations are: $x(\theta)= x_c+ R\cos(\theta)$, $y= y_c + R\sin(\theta)$ where $\theta \in [0,2\pi)$. Mapping the parametric angle θ to the normalized parameter $\xi \in [0,1]$. Hence, $\theta= 2\pi\xi$. The above equations reduce to: $x(\xi)= R\cos(2\pi\xi)$, $y(\xi)= R\sin(2\pi\xi)$.This can finally be represented as $F(\xi,1) =[x(\xi,1), y(\xi,1)] = [R\cos(2\pi\xi), R\sin(2\pi\xi)] =[20\cos(2\pi\xi), 20\sin(2\pi\xi)]$.

c. The left Boundary: The left boundary is cut connecting the top and the bottom boundaries, parameterized by η.

$F(0, \eta) = (1-\eta)\,(F(\xi,0) \mid \xi=0) + \eta F(\xi,1) \mid \xi=0$

Hence $F(0, \eta) = (1-\eta)[a,0] + \eta[R,0] = (1-\eta)[1,0] + \eta[20,0]$

d. The right boundary: The left boundary is cut connecting the top and the bottom boundaries, parameterized by η.

$F(1, \eta) = (1-\eta)\,(F(\xi,0) \mid \xi=1) + \eta F(\xi,1) \mid \xi=1$

Hence $F(1, \eta) = (1-\eta)[a,0] + \eta[R,0] = (1-\eta)[1,0] + \eta[20,0]$

Here, $P_\xi[F(\eta)] = (1-\xi)F(0, \eta) + \xi F(1, \eta)$ and $P_\eta[F(\xi)] = (1-\eta)F(\xi,0) + \eta F(\xi,1)$

The bilinear interpolant P (ξ, η) is generated by the sum $P_\xi[F(\eta)] + P_\eta[F(\xi)]$

P (ξ, η) = (1- ξ) F (0, η) + ξ F (1, η) + (1- η) F(ξ,0) + η F(ξ,1) ……………. (1)

The product projection is given as:

$P_\xi P_\eta(F) = \sum_{i=0}^{1} \sum_{j=0}^{1} \emptyset_i(\xi)\varphi_j(\eta)\, F(\xi_i, \eta_j)$

Here $\emptyset_0(\xi)$=1- ξ, $\emptyset_1(\xi)$= ξ, $\varphi_0(\eta)$=1- η and $\varphi_1(\eta)$= η

Hence $P_\xi P_\eta$(F)= (1- ξ) (1- η) F (0,0) + (1- ξ) (η) F (0,1) + (ξ) (1- η) F (1,0) + (ξ)(η) F (1,1)...(2)

Hence F (ξ, η) = $P_\xi[F(\eta)] \oplus P_\eta[F(\xi)] = P_\xi[F(\eta)] + P_\eta[F(\xi)]- P_\xi P_\eta(F)$

= (1- η) F(ξ,0) + η F(ξ,1) +(1- ξ) F (0, η) + ξ F (1, η)- ((1- ξ) (1- η) F (0,0) + (1- ξ) (η) F (0,1) + (ξ) (1- η) F (1,0) + (ξ)(η) F (1,1))…………(3)

Now applying boundary conditions to equation (1)

P (ξ, η) = (1- ξ) ((1- η) [a,0] + η [R,0]) + ξ ((1- η) [a,0] + η [R,0])+ (1- η) [acos 2π ξ, bsin 2π ξ] + η [Rcos 2π ξ , Rsin 2π ξ]

= (1- η) [acos 2π ξ, bsin 2π ξ] + η [Rcos 2π ξ , Rsin 2π ξ]+ [(1- η) a + ηR, 0] …… (4)

Similarly applying boundary condition to equation (2) for $P_\xi P_\eta$(F)

$P_\xi P_\eta$(F)= (1- ξ) (1- η) F (0,0) + (1- ξ) (η) F (0,1) + (ξ) (1- η) F (1,0) + (ξ)(η) F (1,1)

Here F (0,0) =[a,0], F (0,1) =[R,0], F (1,0) =[a,0] and F (1,1) =[R,0] are obtained after substituting the corner condition in parametric equations. Substituting the conditions in above equation.

$P_\xi P_\eta$(F)= (1- ξ) (1- η) [a,0] + (1- ξ) (η) [R, 0] + (ξ) (1- η) [a,0] + (ξ)(η) [R, 0]

The above expression becomes

$P_\xi P_\eta$(F) = [(1- η) a + ηR, 0] …………… (5)

Substituting equations (4) and (5) in equation (3) the resulting equation becomes

F (ξ, η) = (1- η) [acos 2π ξ, bsin 2π ξ] + η [Rcos 2π ξ , Rsin 2π ξ]

Thus F (ξ, η) = [x (ξ, η), y (ξ, η)] = [(1- η) a cos 2π ξ + ηRcos 2π ξ, (1- η) b sin 2π ξ + ηR sin 2π ξ)] ……………………(6) is the required final TFI equation.

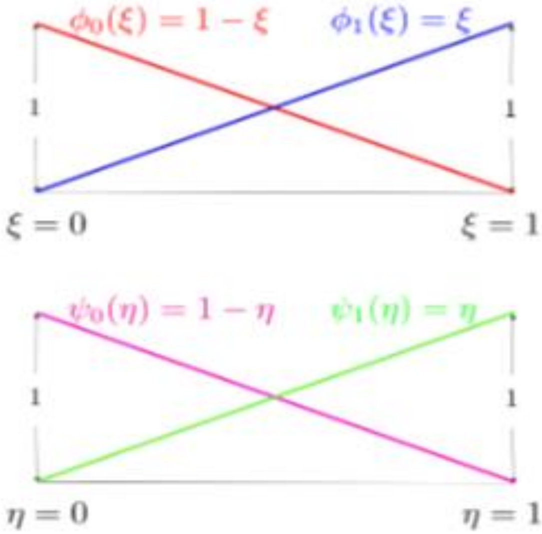
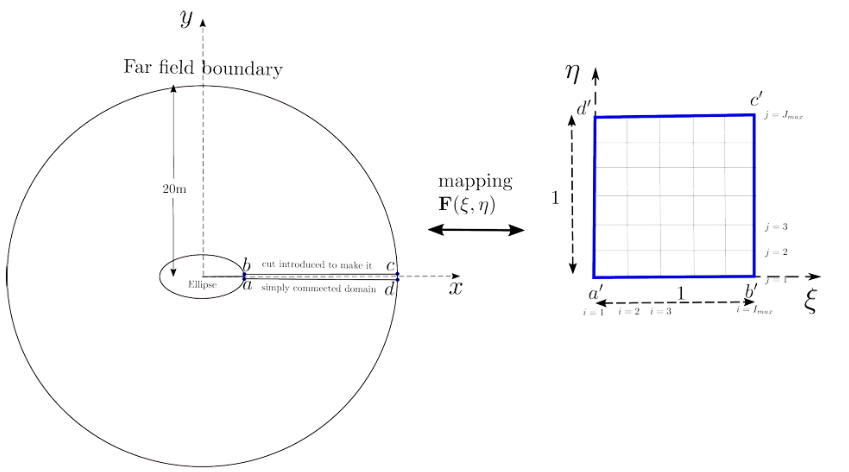


Fig1: Variation of ξ and η



(i) Physical Domain (ii) Computational Domain

Fig2: Grid Mapping used circular cylinder

2.Results

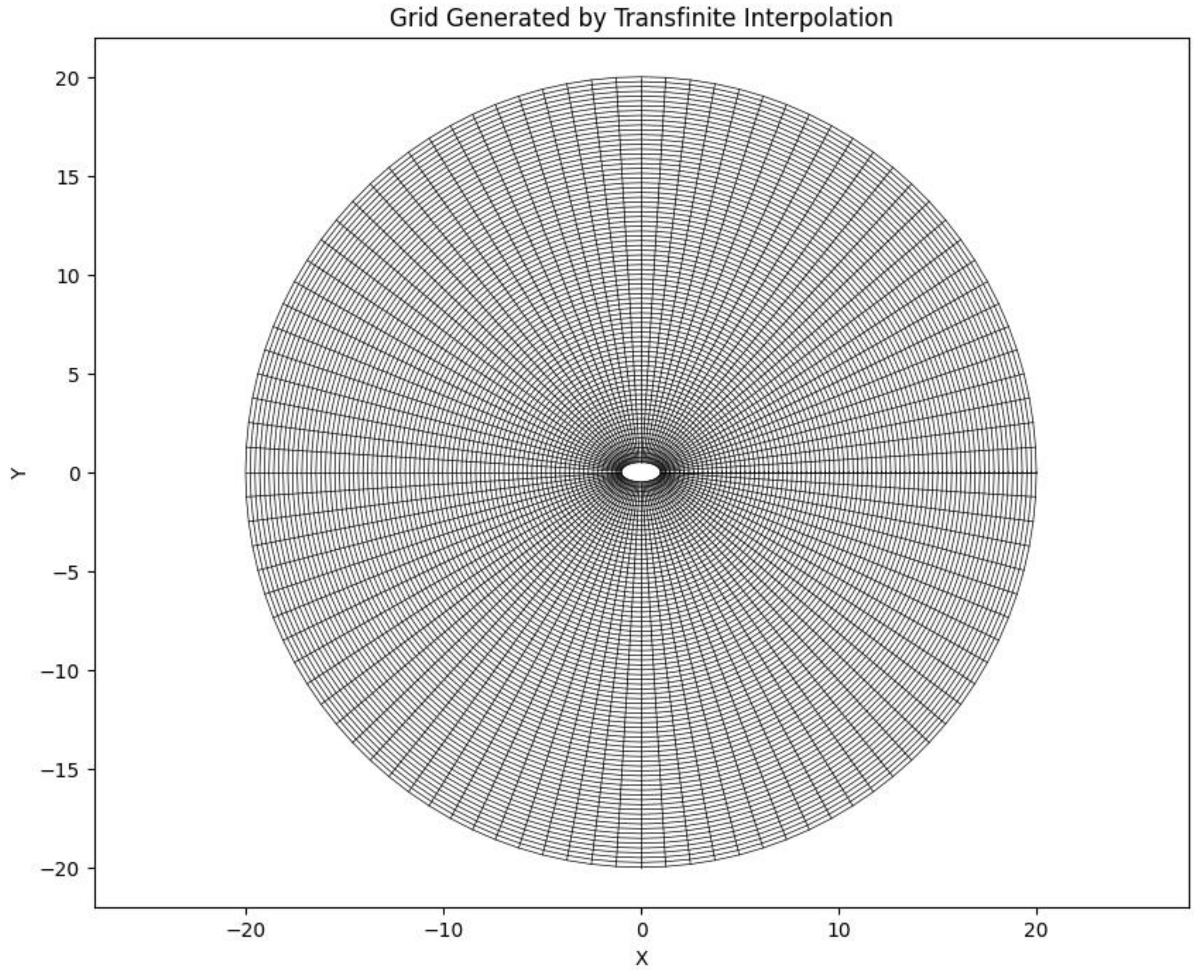


Fig3: Grid in physical domain generated by transfinite interpolation method

The above grid is required grid which is 101 × 81 grids in the physical domain which was obtained using Transfinite Interpolation method.

3.References

i) Assignment-2: Grid generation using Transfinite interpolation (TFI) Method by J C Mandal

ii) Transfinite Interpolation (TFI) Method for Algebraic Grid Generation by J C Mandal

4. Appendix

```
import numpy as np
import matplotlib.pyplot as plt

# Provided parameters in the problem statement
a = 1.0  # Semi-major axis of the ellipse(m)
b = 0.5  # Semi-minor axis of the ellipse(m)
R = 20.0  # Far-field boundary distance/Radius of the circle(m)
N_xi = 101  # Number of grid points in xi-direction
N_eta = 81  # Number of grid points in eta-direction

# Generating the computational grid
xi = np.linspace(0, 1, N_xi)
eta = np.linspace(0, 1, N_eta)
XI, ETA = np.meshgrid(xi, eta)

# Defining  the boundaries and boundary condition
def bottom_boundary(xi):
    """Bottom boundary  is an ellipse"""
    theta = 2 * np.pi * xi
    return a * np.cos(theta), b * np.sin(theta)


def top_boundary(xi):
    """Top boundary is a circle which acts as far-field"""
    theta = 2 * np.pi * xi
    return R * np.cos(theta), R * np.sin(theta)


# Transfinite Interpolation
def transfinite_interpolation(xi, eta):
    """Transfinite interpolation to generate grid points"""
```

```
    # Bottom boundary contribution
    F1_x, F1_y = bottom_boundary(xi)
    # Top boundary contribution
    F2_x, F2_y = top_boundary(xi)

    # TFI formula after substituting boundary conditons and simplification
    x = (1 - eta) * F1_x + eta * F2_x
    y = (1 - eta) * F1_y + eta * F2_y

    return x, y

# Generate the physical grid
X, Y = transfinite_interpolation(XI, ETA)

# Plotting the grid
plt.figure(figsize=(10, 8))
plt.plot(X, Y, 'k', linewidth=0.5)
plt.plot(X.T, Y.T, 'k', linewidth=0.5)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Grid Generated by Transfinite Interpolation')
plt.axis('equal')
plt.show()
```