Proceedings of Computational Heat and Fluid Flow
ME6151
April 30, 2022, IIT Madras, TN, India

AE21M007

FVM SOLUTION FOR STEADY STATE HEAT DIFFUSION IN A 2D UNSTRUCTURED, NON-UNIFORM, NON-ORTHOGONAL MESH WITH CAVITIES

Nachiketa Narayan Kurhade
Graduate Student
Department of Aerospace Engineering
Indian Institute of Technology Madras.

ABSTRACT

Flow straighteners such as honeycomb grid are used to improve flow quality in ducts. The necessity of rigorous structural and thermal design of these flow straighteners becomes important for their integration in mobile utilities of aerospace engineering where saving weight is one of the topmost priorities. Since these grids can have complex geometries, unstructured, non-uniform, non-orthogonal meshes provides a better formulation of the discrete steady state heat diffusion problem than their structured, uniform, orthogonal counterparts. All such problems require identification of outer(edge) and inner(cavity) boundaries which then will have their respective boundary conditions with the constraint that a cell vertex cannot simultaneously be on the inner and outer boundaries. This paper aims to develop a FVM solver that addresses all the above-mentioned requirements in a scalable way. Results produced by this solver are validated through the analytical solution for 1D steady state heat diffusion equation over a disk.

NOMENCLATURE

Alphabets	
A	Area vector
J	Jacobian
S	Source term
T	Temperature
V	Volume
b	Consolidated source terms
c	Constant of integration
e	Direction vector
i	Unit vector in x direction
j	Unit vector in y direction
k	Index of n-dimensional vector
n	Coordinate/vector in normal direction
x	x-coordinate
y	y-coordinate
Greek symbols	
∇	Nabla operator
Γ	Diffusion coefficient
Φ	Scalar quantity one is solving for
ξ	Direction pointing towards next cell centroid
η	Direction tangential to cell face
Δ	Change in quantity/ small quantity
Subscripts	
f	Face index
c	Constant variable
P	Pth cell
x	Partial differential with respect to x
y	Partial differential with respect to y
ξ	Partial differential with respect to ξ
η	Partial differential with respect to η
nb	Neighbor
1	Neighbor cell/index for integral constants
0	Current cell index
b	Boundary cell
i	Lower limit of the integral
Units	
W	Watts
K	Kelvin
m	Meters

INTRODUCTION

Most commercial FVM solver packages out there provide graphical user interface. GUI makes them user friendly, but at the same time most do not provide basic control over how the data structures are created thus forcing the user to make approximations while implementing complex boundary conditions or let us say in our case, most make it tedious as well.

Granted, the user needs to be familiar enough with the code to know how to make these custom adjustments, but given these adjustments are really not that complex, we are in need for solver that allows the user to tweak the code to suit his requirements.

There are open-source solvers which allow to make these changes too, but it caught my attention that over time these solvers have become more and more complex with clever implementation of interconnected data structures and intricate algorithms that make end user go through quite some documentation before one even starts defining the problem.

This paper aims to address this problem through development of a steady state heat diffusion solver which works for an unstructured grid, that has been thought to be difficult to implement but a must have feature if one desires to solve complex geometries with complex boundary conditions.

As an example of such a problem case, we have the problem of 2-D heat diffusion over grids that are used as flow straighteners. These flow straighteners are required to have thin walls to minimize the head loss of flow passing through them[1]. At the same time, these grids in case of a heated flow, (gas turbine combustion chamber exits) due to absence of any immediate cooling solution, which is usually the edge itself, need to be thick enough to allow heat conduction to the edge. This optimization problem can be solved if one is able to find out the temperature distribution over the grid for the given material and flow properties. The timescales and length scales of this problem make this a two-dimensional problem.

A solver aimed to obtain this kind of temperature distribution while having the user-friendly features discussed earlier essentially needs to distinguish outer edge boundaries from inner edge boundaries and thus provide an easy way to implement these boundary conditions without having to visit every single one of them.

For the sake of simplicity, the solver has been limited to meshes formed using triangular elements only. On a face of an element in the mesh, either of the Dirichlet or Neumann boundary condition can be implemented. Least squares method for gradient calculation are recommended to reduce errors observed due to skewness of the mesh. While the solver is very scalable, serial implementation of the program and Gauss Seidel iterative scheme used to solve the equations can make it slow to converge on solution for finer grids.

GOVERNING EQUATIONS

Finite volume methods are used with their control volume approach to discretize equations over unstructured meshes [2,3]. General scalar transport equation for steady state diffusion case is reduced to,

$$\nabla \cdot (\Gamma \nabla \Phi) + S_\Phi = 0 \tag{1}$$

Integrating Eq. (1) over a control volume,

$$\int_V \nabla \cdot (\Gamma \nabla \Phi)\, dV + \int_V s_\Phi \, dV = 0 \tag{2}$$

Applying the Gauss Divergence theorem,

$$\int_{CS} (\Gamma \nabla \Phi) \cdot dA + \int_V s_\Phi \, dV = 0 \tag{3}$$

Taking the linear profile assumption for the source term,

$$\sum_f (\Gamma \nabla \Phi)_f \cdot A_f + (S_c + S_P \Phi_P)\Delta V = 0 \tag{4}$$

Mesh

Equation (4) can be applied on any kind of mesh, be it a structured or unstructured, uniform or non-uniform, orthogonal or non-orthogonal, and to any kind of element, triangular, square, etc. It is also valid for all 1D, 2D and 3D formulations. However, the solver I have developed handles triangular elements only. On the other hand, it can handle the most general case of unstructured, non-uniform, non-orthogonal mesh.

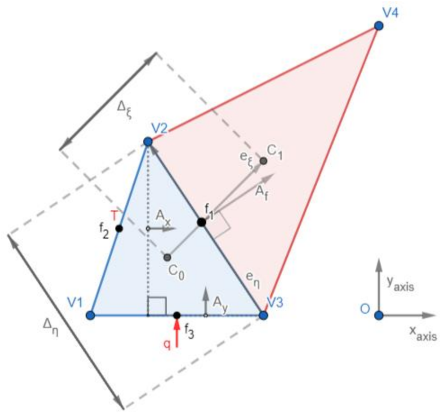

FIGURE 1. Control volume geometry

Without loss of generality, one can write,

$$\nabla \Phi = \frac{\partial \Phi}{\partial x} i + \frac{\partial \Phi}{\partial y} j \tag{5}$$

Changing coordinates x, y to new cell center based coordinate system ξ, η as shown in Fig. 1. one gets,

$$\Phi_\xi = \Phi_x x_\xi + \Phi_y y_\xi \tag{6}$$

Area of the face V_2, V_3 can thus be broken into,

$$A_f = A_x i + A_y j \tag{7}$$

Substituting Eq. (5), Eq. (6), and Eq. (7) in Eq. (4) one gets,

$$\sum_f (\Gamma \nabla \Phi)_f \cdot A_f = \Gamma_f \left(\frac{A_x y_\eta - A_y x_\eta}{J}\right) \Phi_{\xi_f} + \Gamma_f \left(\frac{A_y x_\xi - A_x y_\xi}{J}\right) \Phi_{\eta_f} \tag{8}$$

Where $J = x_\xi y_\eta - y_\xi x_\eta$

With geometric simplifications from Fig. 1. One can reduce Eq. (8) to,

$$\sum_f (\Gamma\nabla\Phi)_f \cdot A_f = \Gamma_f \left(\frac{A_f \cdot A_f}{A_f \cdot e_\xi}\right) \Phi_{\xi_f} - \Gamma_f \left(\frac{A_f \cdot A_f}{A_f \cdot e_\xi}\right) (e_\eta \cdot e_\xi) \Phi_{\eta_f} \quad (9)$$

First term in Eq. (9) is called as the primary gradient term while the second term is called as the secondary gradient term. For orthogonal meshes, e_ξ, e_η are perpendicular, hence secondary gradient for orthogonal meshes becomes zero. Secondary gradient term is also treated as a source term (the one whose values are calculated based on the previous iterations) in the iterative solvers, hence it is a source of error in the solution. Orthogonality in unstructured meshes is thus desired.

Secondary Gradient

The secondary gradient calculation requires us to have Φ stored at cell vortices. This is an inefficient method as it requires large system memory for polygonal cells with more sides. Instead, one can store Φ at cell centroids and use the following relations to calculate secondary gradient.

$$(Total\ Diffusion\ Flux)_f = (PG)_f + (SG)_f \quad (10)$$

$$(Total\ Diffusion\ Flux)_f = \Gamma_f (\nabla\Phi)_f \cdot A_f \quad (11)$$

$$(PG)_f = \Gamma_f \left(\frac{A_f \cdot A_f}{A_f \cdot e_\xi}\right) \Phi_{\xi_f} = \frac{\Gamma_f}{\Delta\xi} \left(\frac{A_f \cdot A_f}{A_f \cdot e_\xi}\right) (\nabla\Phi)_f \cdot e_\xi \Delta\xi \quad (12)$$

$$(SG)_f = \Gamma_f (\nabla\Phi)_f \cdot A_f - \frac{\Gamma_f}{\Delta\xi} \left(\frac{A_f \cdot A_f}{A_f \cdot e_\xi}\right) (\nabla\Phi)_f \cdot e_\xi \Delta\xi \quad (13)$$

Where,
PG : Primary gradient of Φ
SG : Secondary gradient of Φ

Discrete Equation

Taking approximation,

$$\Phi_{\xi_f} = \frac{\Phi_1 - \Phi_0}{\Delta\xi} \quad (14)$$

We arrive at equations of the form,

$$a_P \Phi_P = \sum_{nb} a_{nb} \Phi_{nb} + b \quad (15)$$

Where,

$$a_{nb} = \left(\frac{\Gamma_f}{\Delta\xi} \left(\frac{A_f \cdot A_f}{A_f \cdot e_\xi}\right)\right)_{nb} \quad (16)$$

$$a_P = \sum_{nb} a_{nb} - S_P \Delta V \quad (17)$$

$$b = S_c \Delta V + \sum_f (SG)_f \quad (18)$$

Gradient Calculation

Two methods can be used for gradient calculation, Green Gauss Gradient Theorem, and Least Squares Cell-Based algorithm for calculating cell gradients. Both have their own advantages, first one being less computationally expensive and later one being mesh skewness independent. Since only triangular meshes are being handled, least squares method is found to give superior results in a generalized scenario:

1. Green Gauss Gradient Theorem:

$$(\nabla\Phi)_0 = \frac{1}{\Delta V} \sum_f \Phi_f A_f \quad (19)$$

2. Least Squares Cell-Based algorithm:

$$(\nabla\Phi)_0 = (M^T M)^{-1} M^T d \quad (20)$$

Where,

$$M_k = \Delta r_k \quad (21)$$

$$\Delta r_k = (x_k - x_0)\ i + (y_k - y_0)\ j \quad (22)$$

$$d_k = (\Phi_k - \Phi_0) \quad (23)$$

k: index of the neighboring cell.
Φ_k: scalar value at the centroid of the k^{th} neighboring cell.
Δr_k: position vector of k^{th} neighboring cell's centroid w.r.t. 0^{th} cell's centroid.

Once we have the gradient values at cell centroids, one can calculate gradient values at cell faces $(\nabla\Phi)_f$ using linear interpolation.

Boundary Conditions

The solver is coded to handle either Dirichlet or Neumann boundary condition at a cell face as illustrated in Fig. 1.

To implement Dirichlet boundary condition, one can simply initialize scalar boundary values at desired faces and never update them throughout the evolution of the solution which then converges to exact solution of the steady diffusion equation with forementioned boundary conditions.

To implement Neumann boundary condition, since only fluxes (q) are provided at boundaries, one has to update the scalar (Φ) values at respective boundary faces at the end of every iteration and again iterate with these new face values till one achieves convergence. The scalar values are updated using this simple scalar diffusion equation:

$$q = -(\Gamma\nabla\Phi)_b . n \quad (24)$$

$$q = -\Gamma_b \left.\frac{\partial \Phi}{\partial n}\right|_b = -\frac{\Gamma_b(\Phi_0 - \Phi_b)}{\Delta n_b} \tag{25}$$

Where, n is the coordinate normal to the respective boundary face. With only Neumann boundary conditions on all boundaries one can only find solution up to a constant, which means it requires additional information about Φ value at some point in the domain to fix the solution. This can be implemented by initializing a grid point with the desired Φ value and not updating it throughout the evolution of the solution. Care must be taken to avoid unexpected solutions due to Neumann boundary conditions that do not satisfy energy conservation. If this happens one will find this fixed point serving as the source or the sink of energy in the domain. Solution will still be valid solution; it just might not be the solution one's looking for.

MODIFIED JARVIS MARCH ALGORITHM

Outer edge detection for easier implementation of boundary conditions without having to go through each meshing software's documentation on how it identifies boundary is something I looked forward to implement with this solver. I started with simplest division of boundaries. Identify outer boundary, anything that is not an outer boundary is an inner boundary. In most cases these outer and inner boundary types or ways to compute them are inherently different but they can be similar within their own sets. Making this first division is thus important. In conclusion, I want to make a solver that can solve steady state heat diffusion over a domain that has cavities. I want the solver to identify the outer edges of domain as outer boundaries and the rest as inner boundaries.

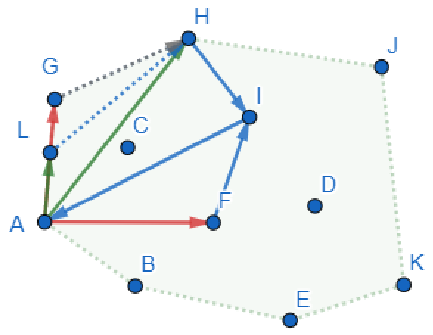

FIGURE 2. The modified Jarvis March algorithm

Jarvis March algorithm has been widely used to identify a convex envelope for a set of given points. I have implemented a modified version of it but it is still limited to convex envelopes. As illustrated in Fig. 2 the algorithm starts with any point (I) that is definitely on the convex hull. Say I start with A. Then I make a rule to pick one more point (II) say I, preferably based on point IDs to make sure I do not miss any. If I want to find the next point in clockwise direction, I pick a random point (III) say F or H in from the pool and check the I-III-II rotation. If the rotation is anti-clockwise (A-F-I), point III i.e. F is not more clockwise to I than II. If the rotation is clockwise (A-H-I), III i.e. H is more clockwise to I than II hence I replace III with II and start the iteration all over again until I run out of all points and still do not get a point more clockwise than the last updated one (G or L). This then becomes my next point on the hull. I start this process all over again till I come back to the first point I started with (it being A in this case) as the next clockwise point for the last point in the hull i.e. B.

There is one special case though which is generally not addressed. From Fig. 2 say I have identified H as the current clockwise point and in the next iteration, depending on how one goes through the points, next point can either be G or L, there is thus a 50% risk of losing L as the next convex hull point, which should not really matter but is important for my solver which wants to identify all the points forming the outer boundary. My modification is to simply check distances in such situation and identify the closest point as the next point on the hull. Once I have identified all the outer boundary points, the rest are simply the inner ones and I can apply separate boundary conditions to the faces that they form.

RESULTS AND DISCUSSION

Six cases of steady state heat diffusion over an annular disk were solved with this solver.

1. *Annular disk with all Dirichlet boundary conditions.*

An annular disk with 1m inner radius and 2m outer radius as shown in Fig. 3. was taken as a test geometry. The domain was converted to a 2D mesh using GMSH meshing software. Use of Frontal-Delaunay 2D algorithm with element size factor 0.1 resulted in 3674 nodes and 7013 elements spanning the entire domain. No source term was considered. Temperature on the inner walls was fixed to 1000K while the temperature on the outer walls was fixed to 100K. Diffusion coefficient was taken to be 10 W/m.

This is a 2D problem for the solver but due to the axisymmetric nature of the domain, solution to this problem is 1D and can be analytically. Governing equation in polar coordinates for this case is:

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r \cdot \Gamma \frac{\partial T}{\partial r}\right) = 0 \tag{26}$$

Which simplifies to,

$$r \cdot \Gamma \frac{\partial T}{\partial r} = c_1 \tag{27}$$

Integrating further from r_i to r one arrives at,

$$T - T_i = \frac{c_1}{\Gamma}\ln\left({}^{r}/_{r_i}\right) \tag{28}$$

Putting in the Dirichlet boundary condition to find the value of c_1 one gets,

$$c_1 = -12984.2554\, WKm^{-1} \tag{29}$$

Thus, the resulting exact solution for this problem is,

$$T = 1000 - \frac{900}{\ln(2)} \ln(r) \tag{30}$$

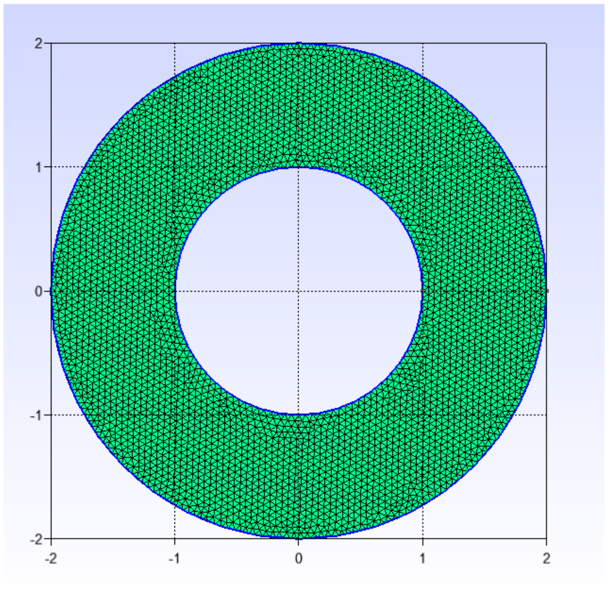

FIGURE 3. Meshed Annular Disk

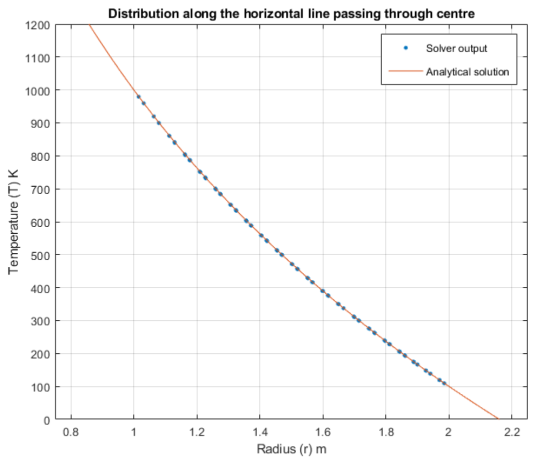

FIGURE 4. Radial distribution of temperature over the annular disk for the test case 1.

Looking at the results (Fig. 4.) from the solver one can easily see that both the solver and the exact solution agree upon the temperature distribution along the radius over the disk. Figure 4 shows the temperature distribution obtained from the solver on the horizontal center line of the domain vs the analytical solution. As one can also notice, since there is no solver domain before 1m and after 2m, no data-points are found in this region. Figure 5 illustrates that this profile is found to be symmetric in all directions while the solver by its very formulation is unaware of any such property of the domain.

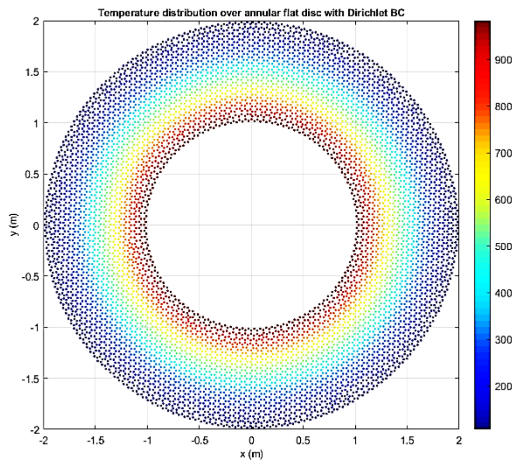

FIGURE 5. 2D distribution of temperature over the annular disk for the test case 1. Color bar is temperature in Kelvins.

2. *Annular disk with all Neumann boundary conditions.*

The same annular disk mesh (Fig. 3.) with the same material properties was taken. Heat flux on the inner walls was fixed to 1000 W/m^2 while the same on the outer walls was fixed to -500 W/m^2. No source term was considered. The fixed point for the solution was taken at x = 1.125m and y = 1.125m while fixing 1000K as its temperature.

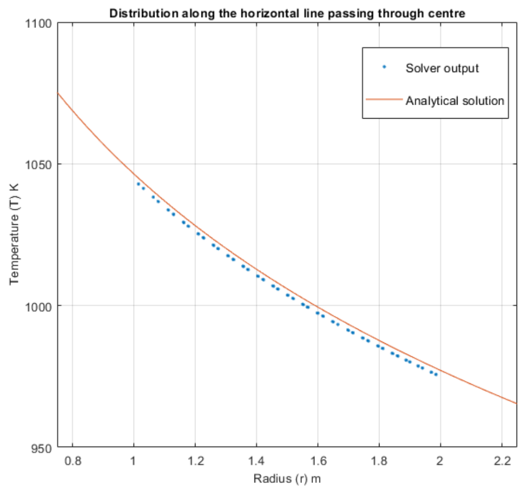

FIGURE 6. Radial distribution of temperature over the annular disk for the test case 2.

Equation (25) and (27) give:

$$c_1 = -1000\, WKm^{-1} \tag{31}$$

Using the fixed point and Eq. 31. In Eq. 28. we get,

$$T = 1000 - 100 \ln\left({}^{r}/_{1.591}\right) \tag{32}$$

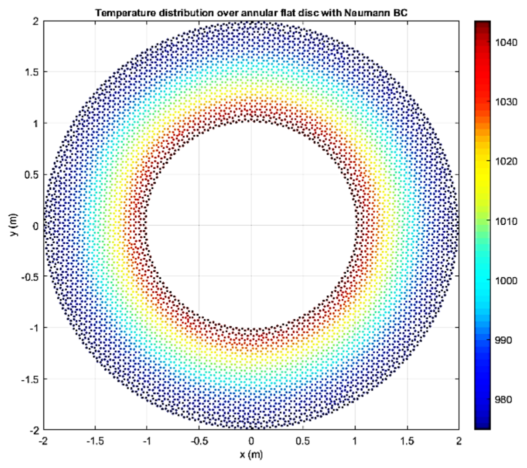

FIGURE 7. 2D distribution of temperature over the annular disk for the test case 2. Color bar is temperature in Kelvins

3. *Annular disk with outer Neumann and inner Dirichlet boundary conditions.*

The same annular disk mesh (Fig. 3.) with the same material properties was taken. Temperature on inner walls was fixed to 500K while the heat flux on the outer walls was fixed to -500 W/m^2. No source term was considered. Equation (25) and (27) give:

$$c_1 = -1000\ WKm^{-1} \tag{33}$$

Using the inner Dirichlet boundary condition and Eq. 33. In Eq. 28. one gets,

$$T = 500 - 100\ln(r) \tag{34}$$

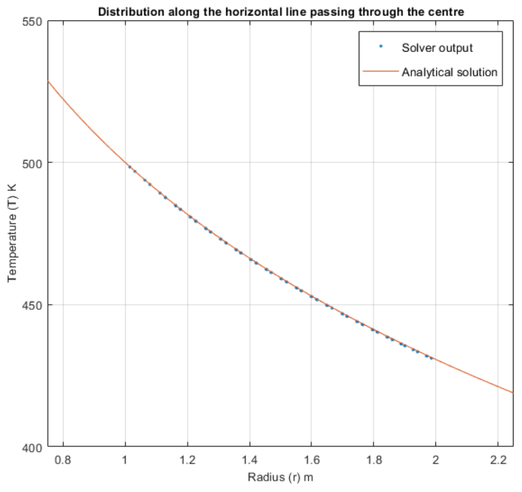

FIGURE 8. Radial distribution of temperature over the annular disk for the test case 3.

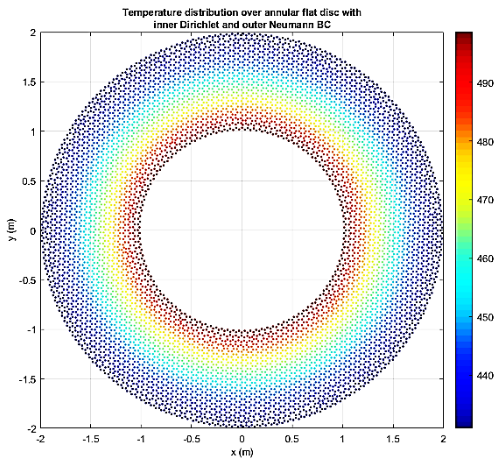

FIGURE 9. 2D distribution of temperature over the annular disk for the test case 3. Color bar is temperature in Kelvins

4. *Annular disk with inner Neumann and outer Dirichlet boundary conditions.*

The same annular disk mesh (Fig. 3.) with the same material properties was taken. Temperature on outer walls was fixed to 500K while the heat flux on the inner walls was fixed to 500 W/m^2. No source term was considered. Equation (25) and (27) give:

$$c_1 = -500\ WKm^{-1} \tag{35}$$

Using the inner Dirichlet boundary condition and Eq. 35. In Eq. 28. Arrives at

$$T = 500 - 50\ln(r/2) \tag{36}$$

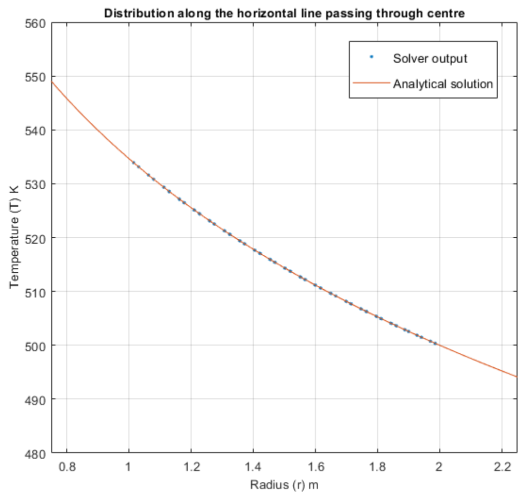

FIGURE 10. Radial distribution of temperature over the annular disk for the test case 4.

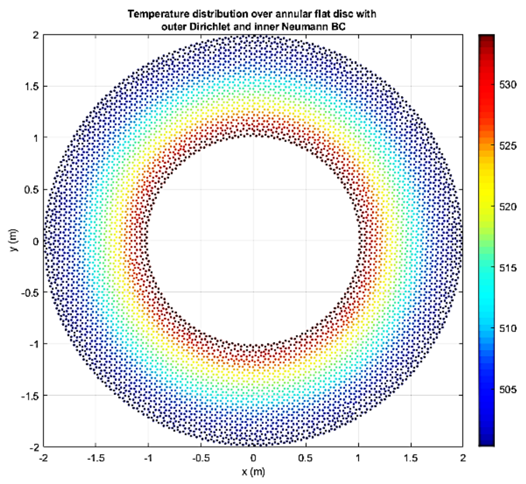

FIGURE 11. 2D distribution of temperature over the annular disk for the test case 4. Color bar is temperature in Kelvins

5. *Disk with heat source and outer Neumann boundary conditions.*

In order to demonstrate that this solver is not limited to implementing boundary conditions as inner and outer boundary conditions but has merely the feature to facilitate the same, we have taken the case of a 2D disk that has constant heat source of 1000 W/m^3 and heat flux observed at the outer boundary is -500 W/m^2. Radius of the disk is 1m. It has been discretized with the same algorithm we used before (case 1) with the same element size factor resulting in an unstructured mesh that has 4729 nodes and 9233 elements spanning the entire domain (Fig. 12.). Since this is an all-Neumann boundary condition case, fixed point is required. For the fixed point, just as we defined earlier x = 0.5m; y=0.5m, and temperature = 1000K. The value of the diffusion coefficient is assumed to be 10W/m.

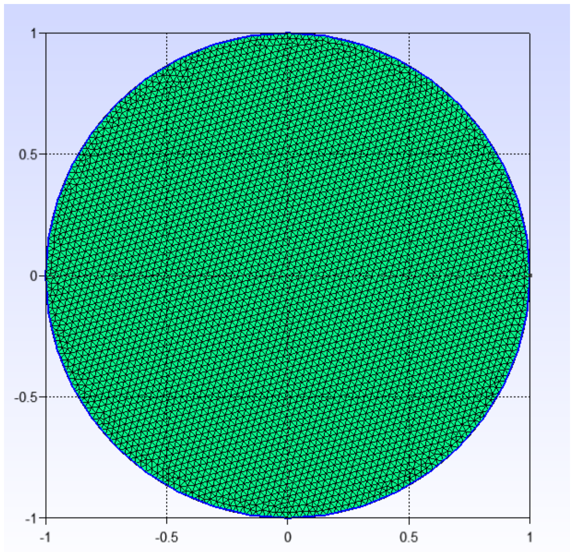

FIGURE 12. Meshed Disk

The governing equation for this problem is given as:

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r \cdot \Gamma \frac{\partial T}{\partial r}\right) + S_c = 0 \tag{37}$$

Simple integration leads to the solutions of the form:

$$\frac{\partial T}{\partial r} = -\frac{S_c r}{2\Gamma} + \frac{c_1}{r} \tag{38}$$

$$T = -\frac{S_c r^2}{4\Gamma} + c_1 \ln(r) + c_2 \tag{39}$$

Putting in the boundary conditions we have $c_1 = 0$ and $c_2 = 1012.5\ K$ hence resulting in the temperature distribution:

$$T = -25r^2 + 1012.5 \tag{40}$$

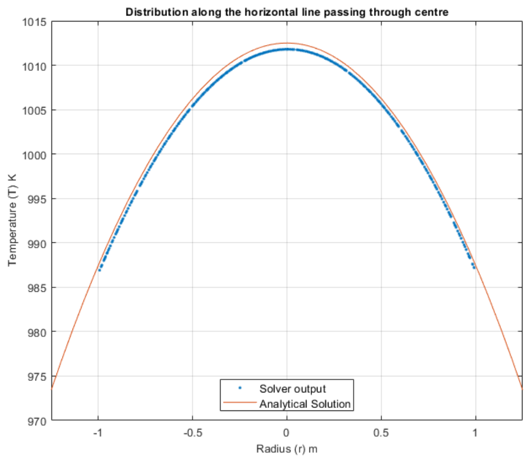

FIGURE 13. Radial distribution of temperature over the disk for the test case 5.

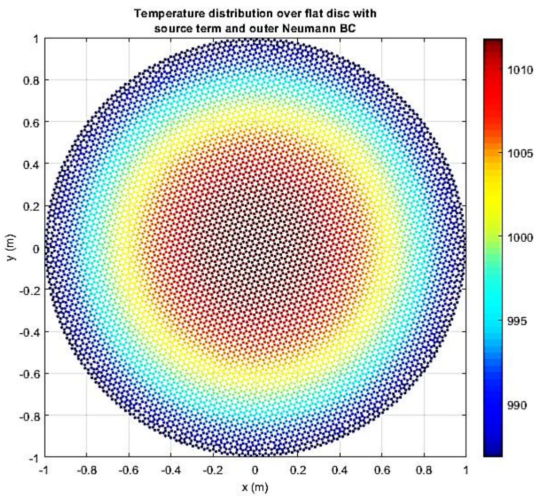

FIGURE 14. 2D distribution of temperature over the disk for the test case 5. Color bar is temperature in Kelvins.

We see that in case 2 and case 5 solver has been missing the exact analytical solution by some error. Given that these two cases were the cases with all Neumann boundary conditions with a fixed point, this error can be attributed to the way this solver implements Neumann boundary conditions numerically since the fixed point is matching with the profile and the point furthest from it is having the highest error.

6. Complex geometry

This is not a case where I validate the results against any literature or analytical solution, rather I want to qualitatively look at the solution produced by the solver for a complex geometry with a highly non-uniform, non-orthogonal unstructured mesh. (Fig. 15) Same algorithm as before has been used for meshing; however, the mesh non-uniformity now is high. Diffusion coefficient for the domain is kept the same.

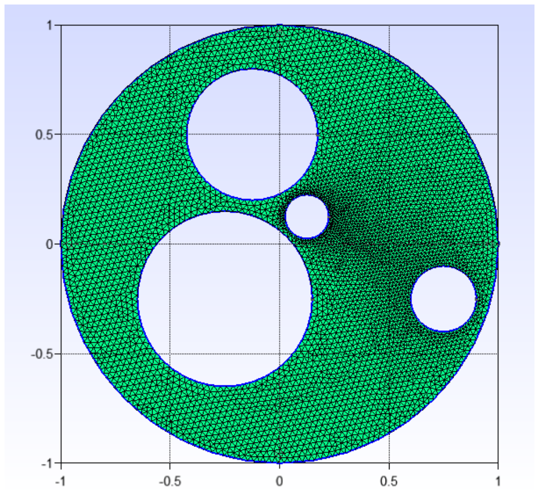

FIGURE 15. Meshed complex annular disk

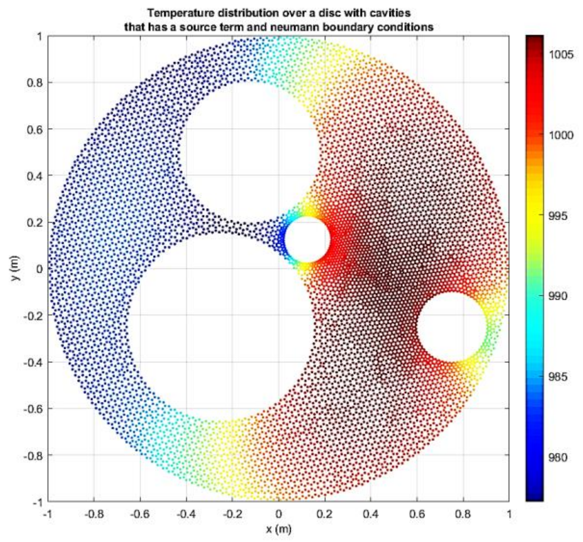

FIGURE 16. 2D Temperature distribution over complex annular disk used in case 6.

Heat generation $S_c = 1000\, W/m^3$ is assumed in the domain. Assuming no heat accumulation, we get 200 W/m^2 and 167.1053 W/m^2 heat flux leaving the outer and inner boundaries respectively. Given this is an all-Neumann boundary conditions case, the fixed point is set up at x = 0.0m, y = -0.75m with temperature 1000K. The outer radius of the disk is 1m whereas the inner circles are of 0.4m, 0.3m, 0.15m, and 0.1m radii.

From Fig. 16. one can notice that the temperature of the disk is less in the areas that have closer proximity to the heat sinks in the form of outflow boundaries. The solution provided by solver hence looks reasonably well behaved.

CONCLUSION

A FORTRAN solver to solve steady state heat diffusion in unstructured meshes made of triangular elements was developed with the theoretical groundwork discussed in the paper. The solver's solutions were compared with the analytical solutions of the cases to validate its accuracy. It was found that where the solutions that involved Dirichlet boundary conditions matched almost exactly, the ones involving all Neumann boundary conditions showed small deviations. All solver solutions were able to match the temperature profile predicted by the analytical solutions.

The solver's feature to be able to distinguish between outer and inner boundary conditions was found useful in reducing the hassle in assigning boundary conditions for complex geometries. The same feature potentially makes solver scalable for problems that involve domains with large number of cavities.

Future work may focus on parallizing the code for better computational performance and reduce the complexity of the modified Jarvis March algorithm as although computed only once, it has been observed to take a significant time to achieve its objective.

REFERENCES

[1] Michal Hrúz, Pavol Pecho, Martin Bugaj, 2020, "Design procedure and honeycomb screen implementation to the air transtport department's subsonic wind tunnel", AEROjournal - ročník 16.; číslo 2/2020, pp 7.

[2] Suhas V Pathankar , 1980, "Numerical Heat Transfer and Fluid Flow", Volume 53 .

[3] H.K: Versteeg and W. Malalasekera, 2005, "Introduction to Computational Fluid Dynamics", volume 44.