iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** I **Month of publication:** January 2024

DOI: https://doi.org/10.22214/ijraset.2024.57936

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue I Jan 2024- Available at www.ijraset.com

3D Heat Conduction Solver for Complex Geometries

Dr. Rambir Bhadouriya
Professor, Maharashtra State Skills University, Mumbai

Abstract: ***The essential pre-requisites for any solver to tackle real life 3D problems are to handle complex geometries and quick turnaround time. In this direction an Unstructured Finite Volume (FVM) based multi-threaded heat conduction solver has been developed to address both the issues respectively. The solver is object oriented and written in Java language. Unstructured Finite Volume method has been utilized for the space discretization and a four stage Runge-Kutta scheme is used for time integration. This solution methodology is highly parallelizable and extensively used in CFD codes. The present solver can handle non-linear steady state and transient 3D thermal problems with Dirichlet, Neumann and surface radiation boundary conditions. The solver has been validated against bench mark problems. In the present work a 3D complex geometry is discretised into 2.5 lakh hexahedrons has been taken as a test case to demonstrate the efficiency of the solver with multi-threading in dual processor machines. The results obtained are presented.***

Keywords: Heat conduction, Object oriented Multi thread, Unstructured Finite Volume method and Runge-Kutta scheme.

I. INTRODUCTION

The governing equations of most of the physical phenomenon are non-linear in nature and are not amenable for obtaining a closed form solution. Hence numerical solution is the only way to circumvent the problem. The numerical solution of a governing equation in a region of interest (computational domain) involves the discretization of the computational domain into entities (elements), decomposition of the spatial and temporal terms in the governing equations into algebraic equations and solving them over the elements.

The elements based on their topology can be triangle, quadrilateral for 2D domains and hexahedron, tetrahedron, prism, pyramid etc. for 3D domains. The discretization methods for the governing equations are Finite Difference, Finite Element, Finite Volume etc [1]. The solution methodology of the governing equations depends on the nature of the physical phenomenon.

It is customary to refer to the computer program which discretizes the computational domain as pre-processor, which solves the governing equations as solver and the one which visualizes the results obtained as post processor.

Of late Unstructured FVM gained popularity over their counterparts because of their adherence to conservativeness and amenability to complex geometries. In this scheme, the solution procedure is carried out in the actual physical domain. Hence, the entire difficulty in numerical simulation relies on the proper discretization of the actual physical domain which in turn depends on the agility of the pre- processor.

II. PROGRAM MODULES

The solver for the thermal analysis solves the energy equation, With the appropriate initial and boundary conditions

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)+q_V=\rho c_p\frac{\partial T}{\partial t}$$

---------- (1)

In the FVM [5] approach the integral form of the equation is used.

$$\iiint\left[\rho c_p\frac{\partial T}{\partial t}-\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)+q_V\right]=0$$

---------- (2)

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue I Jan 2024- Available at www.ijraset.com

By applying Greens theorem

$$\iiint \left[\rho c_p \frac{\partial T}{\partial t} \right] dV = \oint k \nabla T \, dA - \iiint q_V \, dV \quad \text{---------- (3)}$$

Upon discretization and re-arranging

V. dT/dt = R …….. (4)

Where R is the residue

The above discretised equation is to be solved for each element. This can be classified as the following tasks:

- Compute geometry related parameters like Volume, centroid, surface normals, each face centroid for all the elements.
- Compute derivative of the property over each face, which will be required in computing flux across each face of the cell.
- Compute residue for each element.
- Integrate in time.

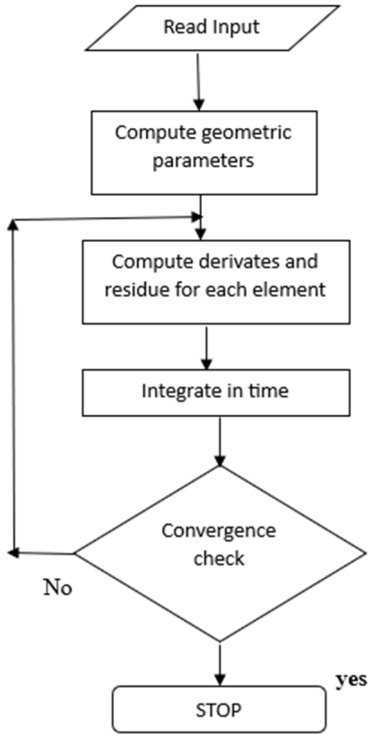


Fig 1: Flow Chart

The flow chart is shown in fig.1. From the above schematic, it is evident that, it is amenable to parallelization. An Object-Oriented approach was adopted in tune with the contemporary programming practice [6]. Parallelization in general involves in identifying the tasks, which can be done independently on an element and performing them simultaneously on different elements using different processors. If all the processors access the same main memory, it is known as Shared Memory parallelization (SMP) and on the other hand if each processor has its own memory, then it is called as Distributed memory (DMP) parallelization.

The latter form of parallelization has a communication overhead, because of the transfer of data among the processors. This is not so in the former as all the processors access the same main memory.

The process of subdividing the tasks and submitting to various processors in SMP is known as multi-threading.

In the present work, JAVA language [3,4] has been used for developing the solver. JAVA has been chosen because of its object-oriented nature and well-defined thread class which resulted in less development time. In the case of parallelization using JAVA threads in an SMP machine, the simplified programming efforts allows to concentrate more on the physics and numerics rather than computer programming.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue I Jan 2024- Available at www.ijraset.com

III. DESCRIPTION OF SOLVER

The solver as described earlier solves 3D energy equation by Unstructured FVM method, it uses Runge-Kutta scheme for time integration and handles hexahedral elements.

The following are the various classes in the solver

- Class Element
- Class Ghost
- Class Node
- Class Solver
- Class Main

A. Class Element

The following are the various methods in the class Element:

1) Constructor: Element (Nodes): To create element object with given attributes like connecting nodes and dependant variables.
2) Time step: Setdt (Element): To compute the time step for given element
3) Compute Residue
a) Setresidue (facenum, Element): To compute the residue of a given element when its neighbour across a given face is an element
b) Setresidue (facenum, Ghost): To compute the residue of a given element when its neighbour across a given face is a ghost

4) Compute Volume: Setvol (): To compute the Volume of a given element
5) Compute minimum edge length: Setdi (): To compute the minimum edge length of each element
6) Compute normals: Setnormals (): To compute the face normals of each element
7) Compute centroids: Setcentroids(): To compute the face centroids and centroid of a given element.

B. Class Node

1) Constructor: Node (coods, nbrs, elsun): To create a node object with given coordinates, count of the elements surrounding it and their numbers
2) Reconstruction: Reconstruction (Elements): To compute the nodal values of the dependant variables from the centroidal values.
3) Class Ghost
a) Constructor

Ghost (Parent, boundary type, Facenum, heatb transfer coeff., Temperature,)

To create a Ghost element with a given element as parent, on a given side, with specified heat transfer coefficient, surrounding temperature.

b) Compute Ghost Centroid

Setghostcentre (Elements)
To compute the centroid of the ghost.

C. Class Solver

1) Solver: Solver (name, start, finish, Element, Node, Ghost) To run the solver thread for a specified number of elements.
2) Run: Run ()
To compute and update the dependant variables

D. Class HexaHeat

Main
Main ()
To handle input / output operations, distribute the elements among the processor, create, run and synchronise the threads.

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue I Jan 2024- Available at www.ijraset.com

IV. RESULTS AND DISCUSSION

A. Validation

The JAVA code developed for validation of 3D unsteady state heat conduction equation is validated against following standard test cases for semi – infinite body (Fig.2) for which analytical solutions are available [2].

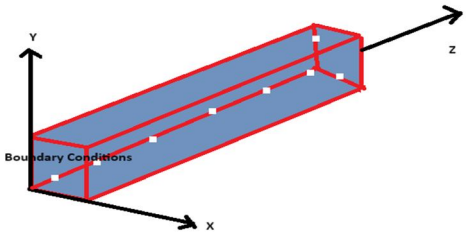


Figure 2: Semi-Infinite body

1) Constant heat flux
2) Constant heat transfer coefficient
3) Constant temperature

The results were compared with the analytical solution. Figs. 3, 4 shows the comparison. The agreement between the prediction and analytical solution paved the way for handling a real-life 3D problem.

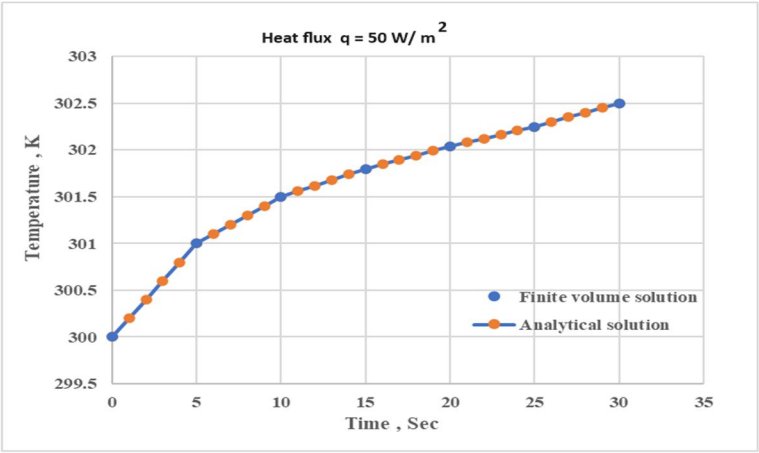


Figure 3: Validation for constant heat flux

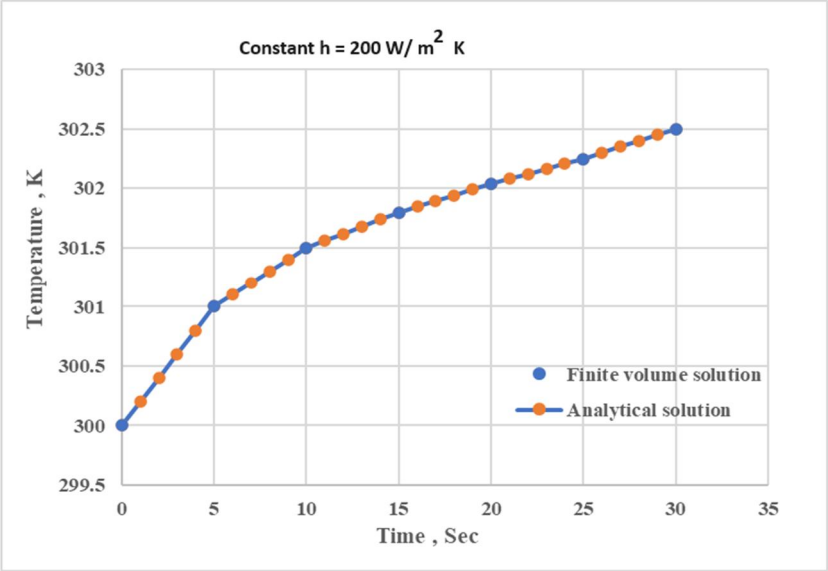


Figure 4: Validation for constant heat transfer coefficient

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue I Jan 2024- Available at www.ijraset.com

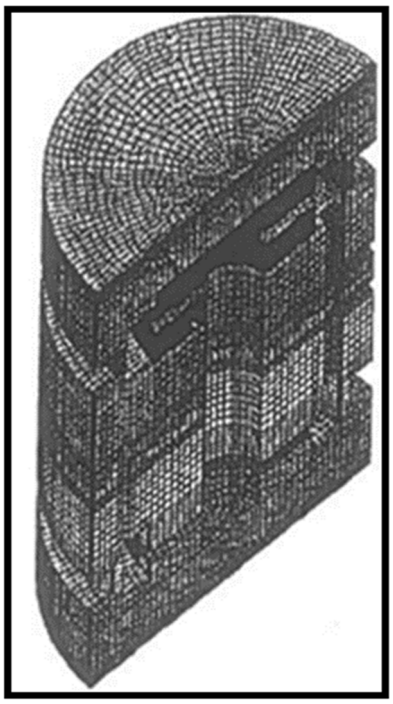


Figure 5. Computational domain for the analysis – different parts of the assembly

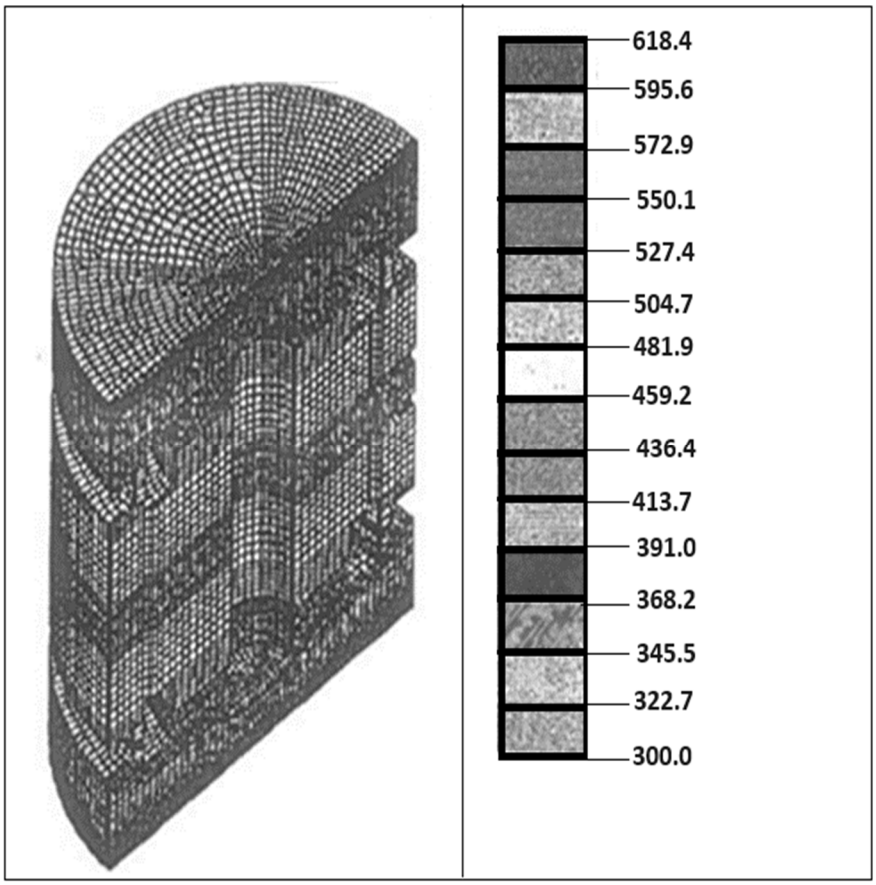


Figure 6: Isothermal Surfaces in the computational domain

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

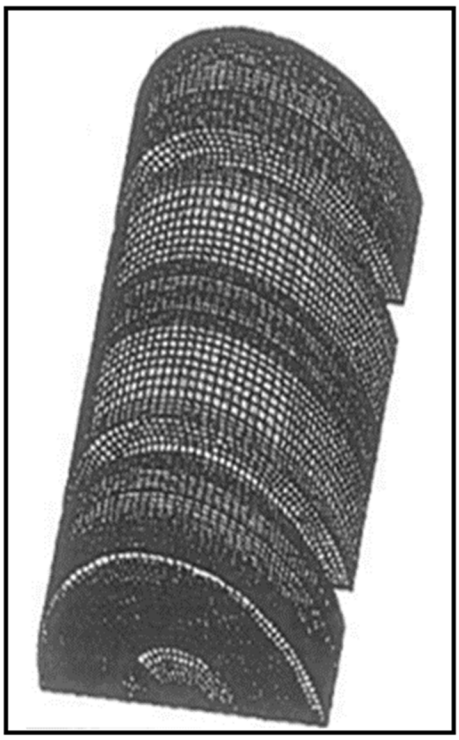


Figure 7. Isothermal surfaces in the computational domain – view 2.

To demonstrate the ability of the solver to handle complex geometries, a valve designed for high temperature and high pressure is modelled to simulate isothermal surface inside the valve under steady state operation. The critical part of the valve is disc region which is analysed in the present problem. The computation domain is shown in Fig 5. The computational domain is discretized into 225075 nodes and 205920 hexahedral elements. The valve is exposed to high temperature on one side and ambient conditions on the other side. Water enters the valve chamber at 30 C with a velocity of 2 m/sec at the inlet region. Convective heat transfer boundary conditions derived from correlations were specified on the coolant contact surfaces. The isothermal surfaces are shown under steady state operation are shown in Fig.6,7.

To demonstrate the efficiency of multi- threading, the problem was first run on a single processor with single thread. Then on a dual processor two threads, three threads etc. were done. The time taken is as tabulated.

No. of Processors	No. of threads	Time taken, Sec
1	1	2280
2	2	1260

The efficiency obtained is 90.47%. All the computations were performed on an Apple Mac G5 dual processor machine with 4GB RAM. With the availability of multi-core multi-processor system this methodology will enable us to considerably bring down the turnaround time for real life 3D problems.

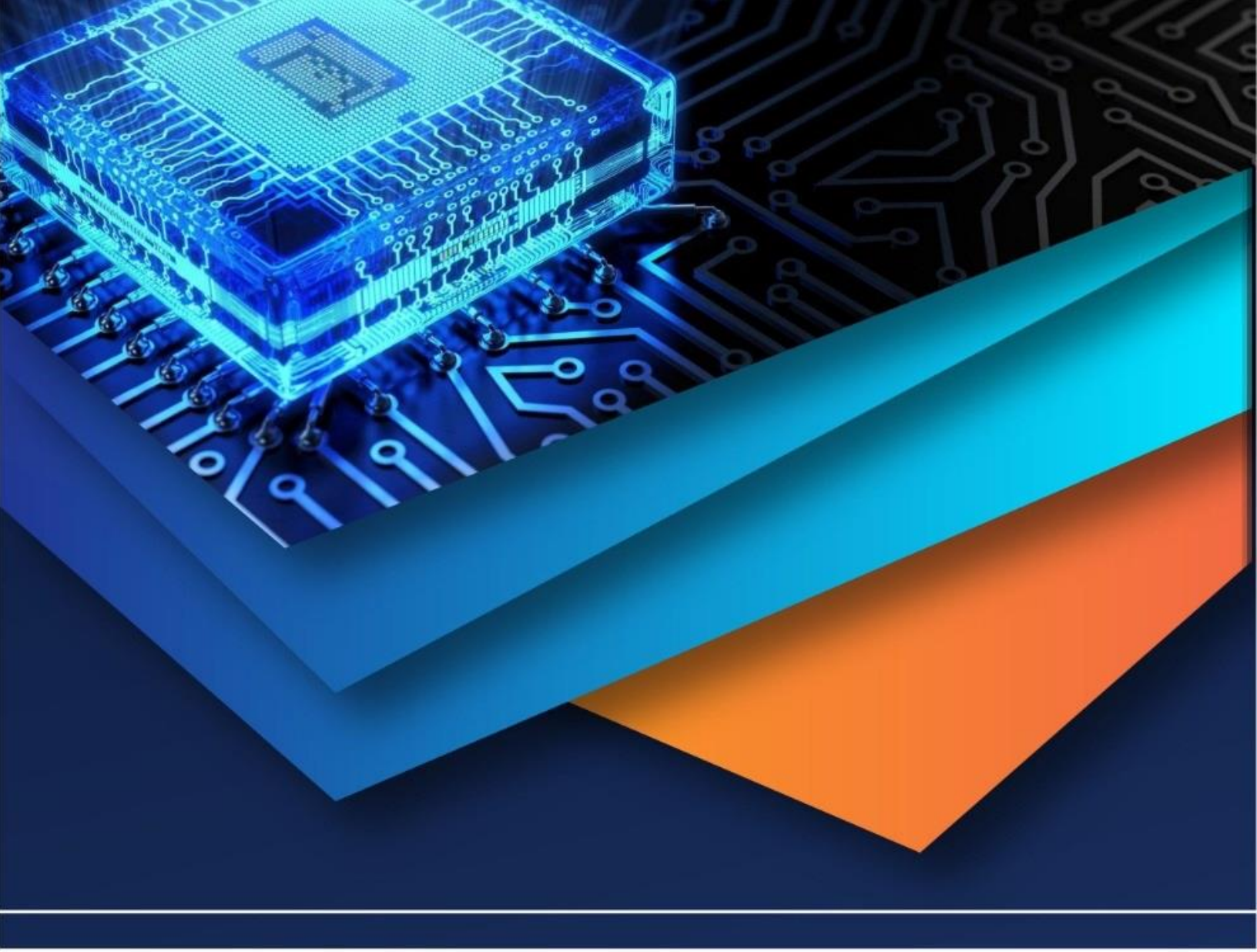
V. FUTURE WORK

1) To add normal radiation also into the code.
2) To increase the efficiency by further optimization of the algorithm.

REFERENCES

[1] Charles Hirsch, Computational Methods in internal and external flows Vol. 1 & Vol. II, John Wiley & Sons, New York, 1992.
[2] Carlslaw, H. S and J.C. Jaeger, Conduction Heat Transfer in Solids, 2nd ed. Oxford University Press, London, 1959.
[3] Herbert Schildt, Java The complete reference, New York, McGraw-Hill,2004.
[4] E. Balagurusamy., Programming with Java A primer, 2nd Ed, Tata McGraw-Hill, NewDelhi, 2005.
[5] H.K. Versteeg and W. Malalasekara, An Introduction to Finite Volume Method, Longman scientific and technical publication, 1995.
[6] A.G. Malan, R.W. Lewis, "On the development of high-performance C++ Object oriented Code with applications to an explicit edge-based fluid dynamics scheme", Computers & Fluids, 33,2004, pp1291-1304.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)