Journal of Mines, Metals and Fuels, *71(12B): 151-160; 2023. DOI: 10.18311/jmmf/2023/45544*

Print ISSN : 0022-2755

Journal of Mines, Metals and Fuels

Contents available at: *www.informaticsjournals.com/index.php/jmmf*

Parallel Finite Element Analysis of 1D and 2D Problems of Heat Distribution in Mine Ventilation Galleries

Ashish Narayan[1], Bhupal Kumar[2]*, Ranjan Kumar[3] and Subodh Kumar[4]

[1]*Department of Mechanical Engineering, Ramgarh Engineering College, Ramgarh – 825101, Jharkhand, India*
[2]*Government Engineering College, Jamui – 811313, Bihar, India; kbhupal246@gmail.com*
[3]*CSIR-Central Institute of Mining and Fuel Research, Dhanbad – 826015, Jharkhand, India*
[4]*SR University, Warangal – 506371, Telangana, India*

Abstract

The analyses of large-scale numerical problems of mining domain such as heat dissipation from rock strata to the ventilation air in large network of underground mine galleries are usually time taking process and require huge computational resources. To overcome this issue, High Performance Computing system (HPC) can be utilized to expedite the numerical analysis process for solving the problems within a reasonable time. In order to utilize HPC, this paper developed parallel finite element codes for solving heat conduction problems of mine galleries. The developed C++ codes of parallel FE are applied in solving 1D and 2D heat conduction problems. For the comparative analysis of 1D and 2D problems, the number of unknown nodes is fixed. It is observed that the computational time reduced when the number of processors increased. On the other hand, the computational time took longer time in 2D problem in comparison to that in 1D. Graphics User Interface (GUI) is also developed in terms of number of nodes of mesh, the boundary temperatures and length of the element. To sum up, the work of this paper reveals that the parallel computing is highly efficient in solving the large scale numerical geo-mining problems.

Keywords: *Computational Time, Iteration, Mines, Parallel Programming*

1.0 Introduction

In current day the computational work is involved more to solve the big domain problem in several fields in engineering. So, for this proposes the resources is available high computing as well as 24x7 hours continuously working computation is required. Therefore, the different computing system is developed for match our recent problem which are getting day by day[1]. In this field more complex and big domain problem its need to improve processing speed therefore more than one processor to facilitate the solution of a particular problem may be termed as parallel processing. Parallel computing is a type of computation where several calculations are carried out simultaneously. In this the problem is divided into discrete pieces of work and solve individual, and each part is further broken down to a series of instructions. The different processors executed instruction from each part at the same time. The basic idea was to use many computer processors simultaneously but present it a single computer interface to a user. These resources may be widely geographically located but shared on computer clusters and presented as a single harmonized computer for solving data intensive and large scale computations. Cluster computing is more like grid computing with the difference being use if cluster rather than grids. Clusters can be distributed or parallel systems that are made up of interconnected computers working together as a single resource[2,3].

*Author for correspondence

Babuska and Elman[4] has discussed the finite element method using high-order and low- order elements some aspects of on parallel computers with direct and iterative solvers for solving the linear systems that occur. They observed that the high order methods are better for achieving the high accuracy for some problem by using series as well as parallel computers. Further, it observed that the direct solvers generalize naturally to methods based on high-order elements, and that direct solvers are adequate for two-dimensional problems, especially for multiple load vectors.

Choporov[5] is discussed the formation of assembly of stiffness matrix and solution of linear equations in finite element method with parallel computing systems. In this paper it is discussed the two method first is Dirichlet problem on the complex domain (gear); Elasticity problem of three-layer shell. Further, it compares the performance of solutions of these problems with one, two and four parallel cores.

Wasy *et al.*[6] was studied numerically for simple and cracked cantilever beams using finite element stiffness method, analytical beam theory, ANSYS and then verified results with code generated in C++ language. It was observed that the finite element method gives less deflection as compared with the results obtained using beam theory for any specific location along span length. MD-Solid software is used to draw bending moment and shear force diagram.

Riaz[7] is developed the finite element domain in C++ and solving the Finite Element problem on shared memory as well as distributed memory architectures. The finite element domain assembles the system of equations, computes its solution, and provides it back to the user through element objects.

Ahmad[8] has discussed to build a Beowulf class PC cluster and then code the required programs using such free and open-source tools as MPI and OpenMP. The FORTRAN is used to program two-dimensional finite element analysis and the same code is then modified for use in a parallel computing environment.

Rauber and Rünger[9] are comparative studied the two different classes of Runge–Kutta (RK) methods providing embedded solutions (i) classical embedded RK methods (ii) iterated RK methods of different order by applying them to ODE systems with different characteristics. It is observed that the embedded RK methods provide lowest runtime in most cases because of their efficient computational organization.

Cruz and Monsivais[10] studied to considering immiscible and incompressible displacement of the two-phase flow model in a homogeneous porous media. This model is solved numerically by using Finite Volume Method (FVM) and compares four numerical schemes for the volumes. It describes briefly how to obtain the mathematical and computational models applying axiomatic formulations and generic programming. Two strategies of parallelization are implemented in order to reduce the execution time. We study distributed (cluster of CPUs) and shared (Graphics Processing Units) memory architectures. It is observed that the GPUs cluster will have better performance than a CPU only cluster.

2.0 Numerical Problems of Heat Distribution in Mine Galleries

2.1 Heat Distribution Phenomena in Mine Galleries

As the mining horizon is shifting towards the greater depth, the heat transfers from the rock mass to the mine working creates problems before the mine management. It has been shown that in deep mines, heat transfer from the rock mass accounts more than 75% of the total mine heat load[11]. The ventilation networks are the interconnections of mine ventilation galleries spread all over the mine and can be of several kilometers in length. The heat transfer from rock mass must be assessed throughout the network of galleries and quantified in order to take precautionary measures in maintaining proper temperature in mine workings. Figure 1 shows the schematic of air flow from intake to exhaust through a mine ventilation network.

The governing equations of 1D and 2D heat conduction from rock mass to mine air can be expressed by the following equations:

$$k\frac{d^2T}{dx^2}+q=0$$

$$k\left(\frac{d^2T}{dx^2}+\frac{d^2T}{dy^2}\right)+q=0$$

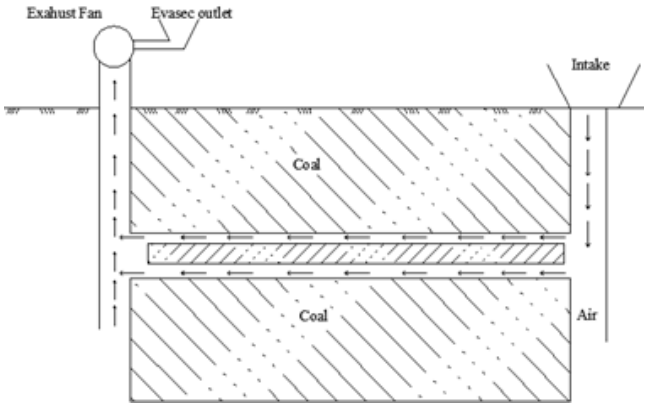


Figure 1. The schematic of a mine ventilation network.

Because, the network length is in kilometers which is very long in compared to the width in meters, this paper formulates finite element models mainly in 1D and 2D domain.

2.2 Formulation of 1D and 2D Model of Heat Distribution

The schematics of the 1D and 2D problem on heat conduction problem showing the boundary condition are depicted in Figure 2. The codes are developed in series as well as in parallel for finding the temperature at the unknown nodes with varying the boundary condition.

The Figure 2(a) is the schematic diagram of 1D problem for temperature distribution of 1m long rod and the Figure 2(b) is the temperature distribution in a 1.5m x 1.5m square slab using a coarse mesh generation.

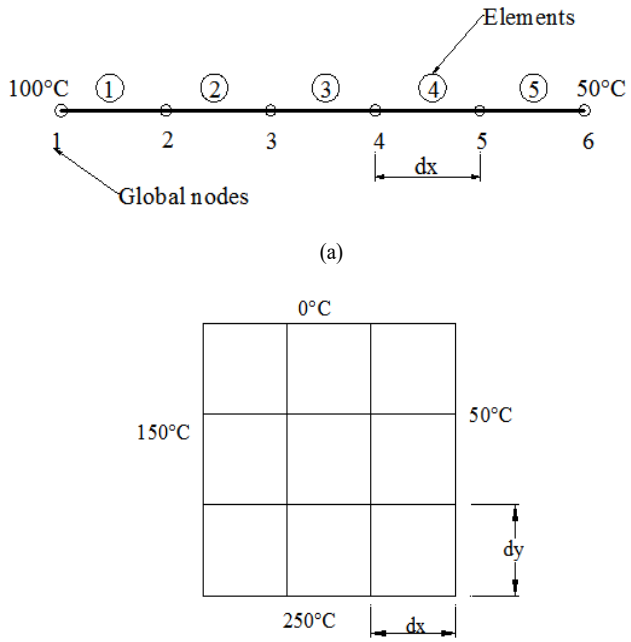


Figure 2. Schematic diagram of **(a)** 1D **(b)** 2D heat conduction problem.

3.0 Methodology for Parallelisation of FEM

3.1 Message Passing Interface

MPI (Message Passing Interface) is an interface to a library comes with bindings for C/C++ or for other languages. It is also a standard library for message passing and programming model for running processes ubiquitous in HPC (High Performance Computing). Additionally, each process has its own data. Moreover, it is a directory of C/**C++** programs which illustrate the use of the MPI for parallel programming and allows a user to write a program in languages such as C, C++, FORTRAN, etc. and carry out a computation in parallel on an arbitrary number of cooperating computers on distributed memory architectures. The MPI facilitates virtual topology, inter-

Table 1. MPI commands used in parallel programming

Commands	Global Actions/ communications
MPI_Init	Initialize the parallel machine
MPI_Finalize	Stop the parallel machine
MPI_Comm_size	How many process are there
MPI_Comm_rank	What is my number
MPI_Send	Sends the exact count of elements
MPI_Receive	receive at most the count of elements
MPI_Reduce	Handle almost all of the common reductions

communication and synchronization of tasks between a set of workers, in a language independent method, with language specific syntax and features. The finite element problems can always be reduced to a system of linear/ non-linear algebraic equations which can be represented in form of matrices.

In principle, the code between MPI_Init() and MPI_Finalize() is executed by all processes. This allows assigning different tasks to different processes by making use of *if*-statement. The complete task is then accomplished by inter process communication.

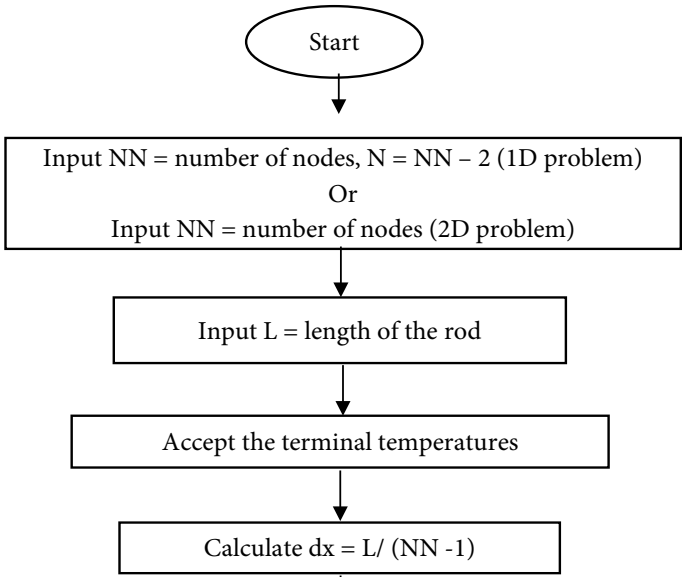
3.2 Implementation of MPI

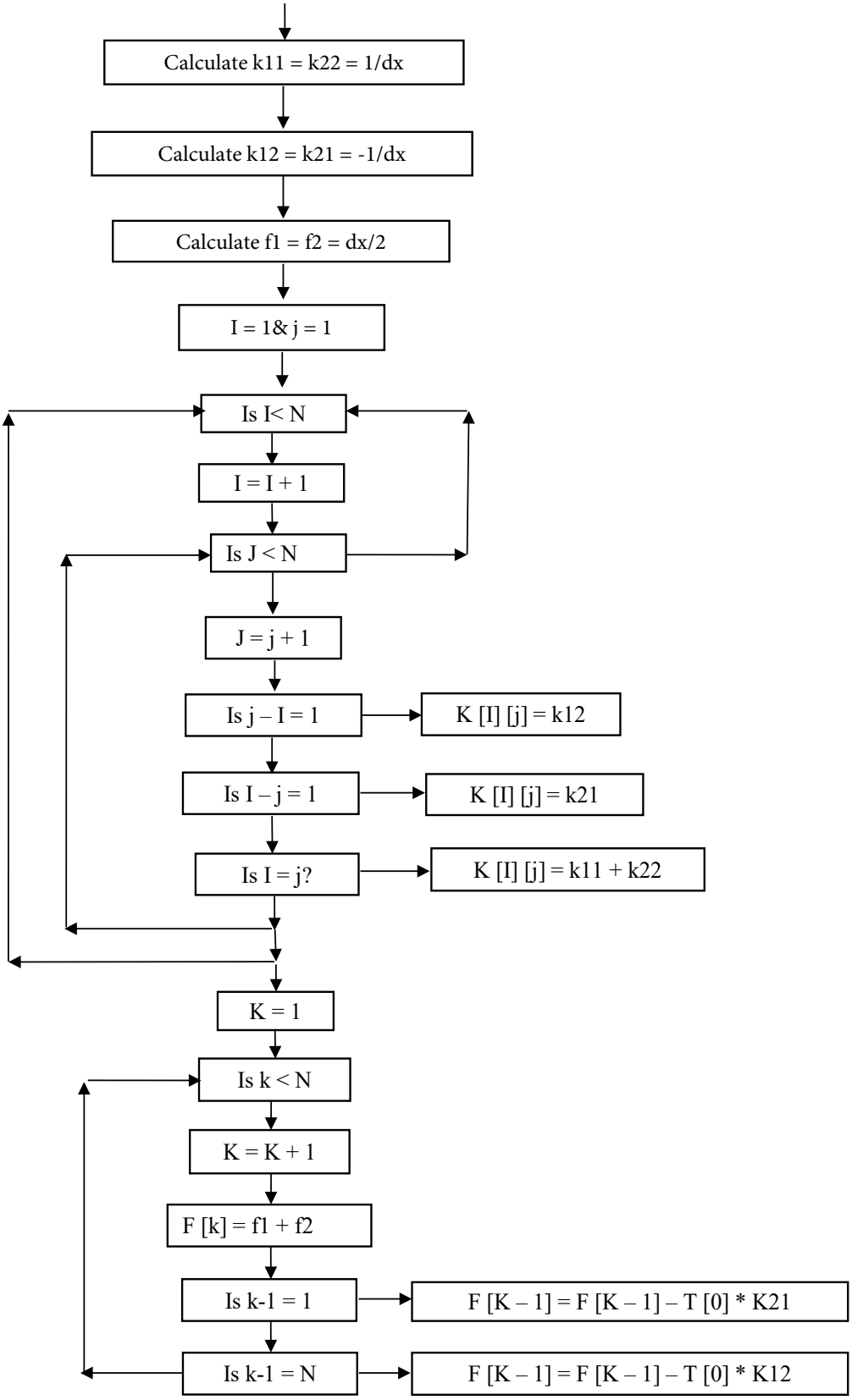
Once the required environment for parallel computing is set up, the program can be made dynamic for variable number of workers. The worker number '0' is called the master. The master worker is the one which sends data to all other workers which are called the workers and then in turn, compiles the result from each of these worker workers to solve the whole problem. In this case, individual rows can be sent the task to each worker for matrix manipulation operations as the calculation of each row is independent of the other. Hence, the code is segmented to be run by the master and worker workers for the distribution of the number of operations required to calculate the inverse of a matrix and product of two matrices. The run time recorded to keep a track of how change in the parameters involved including the number of workers influence it.

The matrix multiplication of the B and C matrices have been split as discussed earlier. There are 256 workers available and any number of these workers can be used for solving our problem. The number of workers can be varied from 2 to 64 eventually. The first worker is called as the master worker. This master worker will be reserved for the pre-calculation of the two matrices before they can be sent for multiplication. The remaining 'N-1' of the 'N' workers is used for splitting the multiplication operation equally among them. Hence the floor value of Rows/ (N-1). The remainder of the number is split amongst the remainder worker workers. This reduces the complexity of our algorithm from 'O' (Rows x Rows) to O (Rows x Rows / N), hence making our code effectively 64 times faster! On real problems where domain size is of the order 10^{18} operations, where the serial code may take weeks to produce the results, our parallel code will run in just a few hours.

4.0 Flow Chat

The flow chart for the 1D and 2D problem for series as well as parallel format is given in Figure 3. The flow chart




Calculate k11 = k22 = 1/dx
Calculate k12 = k21 = -1/dx
Calculate f1 = f2 = dx/2
I = 1& j = 1
Is I< N
I = I + 1
Is J < N
J = j + 1
Is j – I = 1
K [I] [j] = k12
Is I – j = 1
K [I] [j] = k21
Is I = j?
K [I] [j] = k11 + k22
K = 1
Is k < N
K = K + 1
F [k] = f1 + f2
Is k-1 = 1
F [K – 1] = F [K – 1] – T [0] * K21
Is k-1 = N
F [K – 1] = F [K – 1] – T [0] * K12

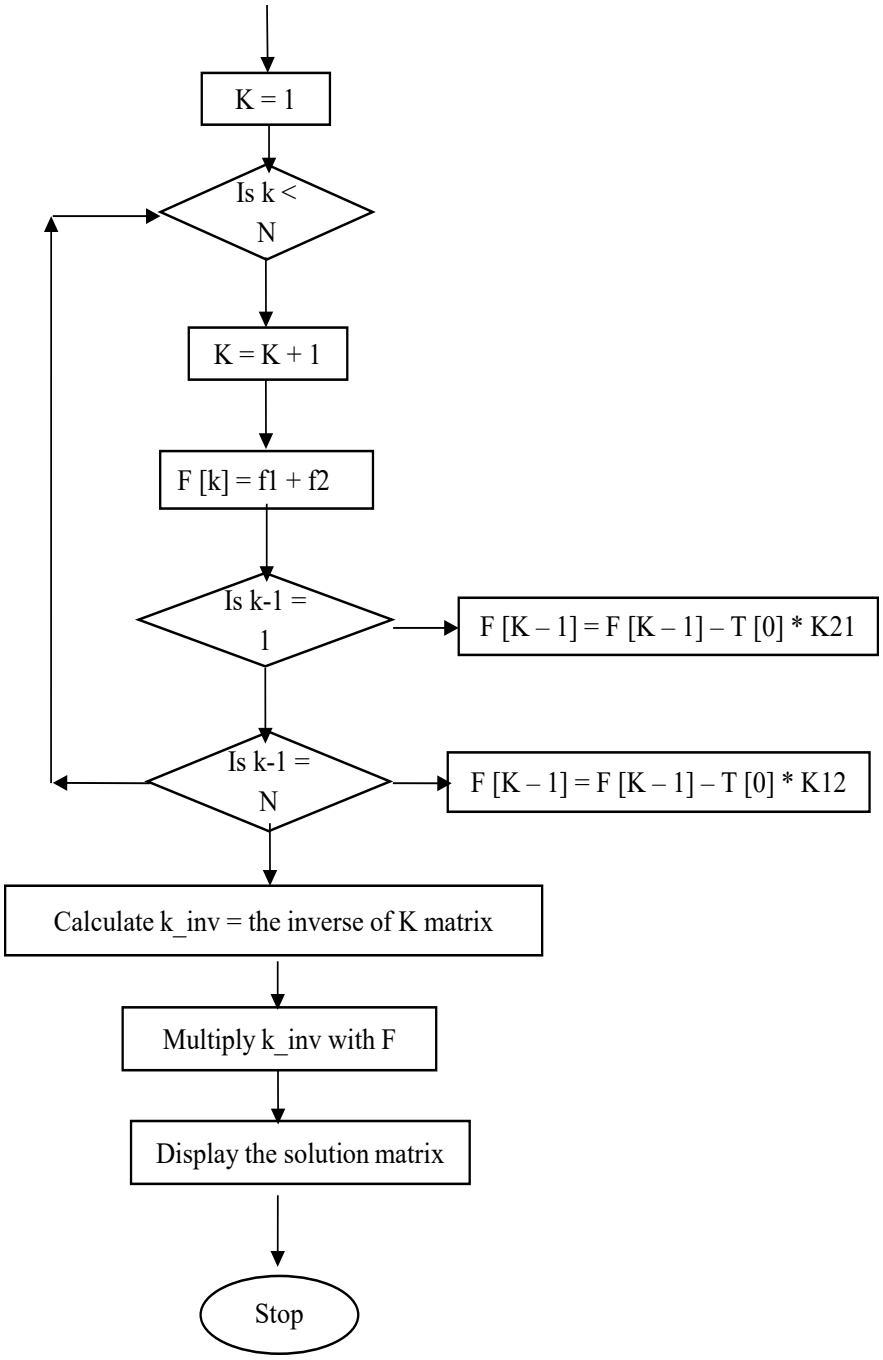


Figure 3. Flow chat of 1D and 2D problems

is provided the detailed about the program step wise and it is a summarized of the code. The flow chat for convert the series program to parallel computing is shown in Figure 4.

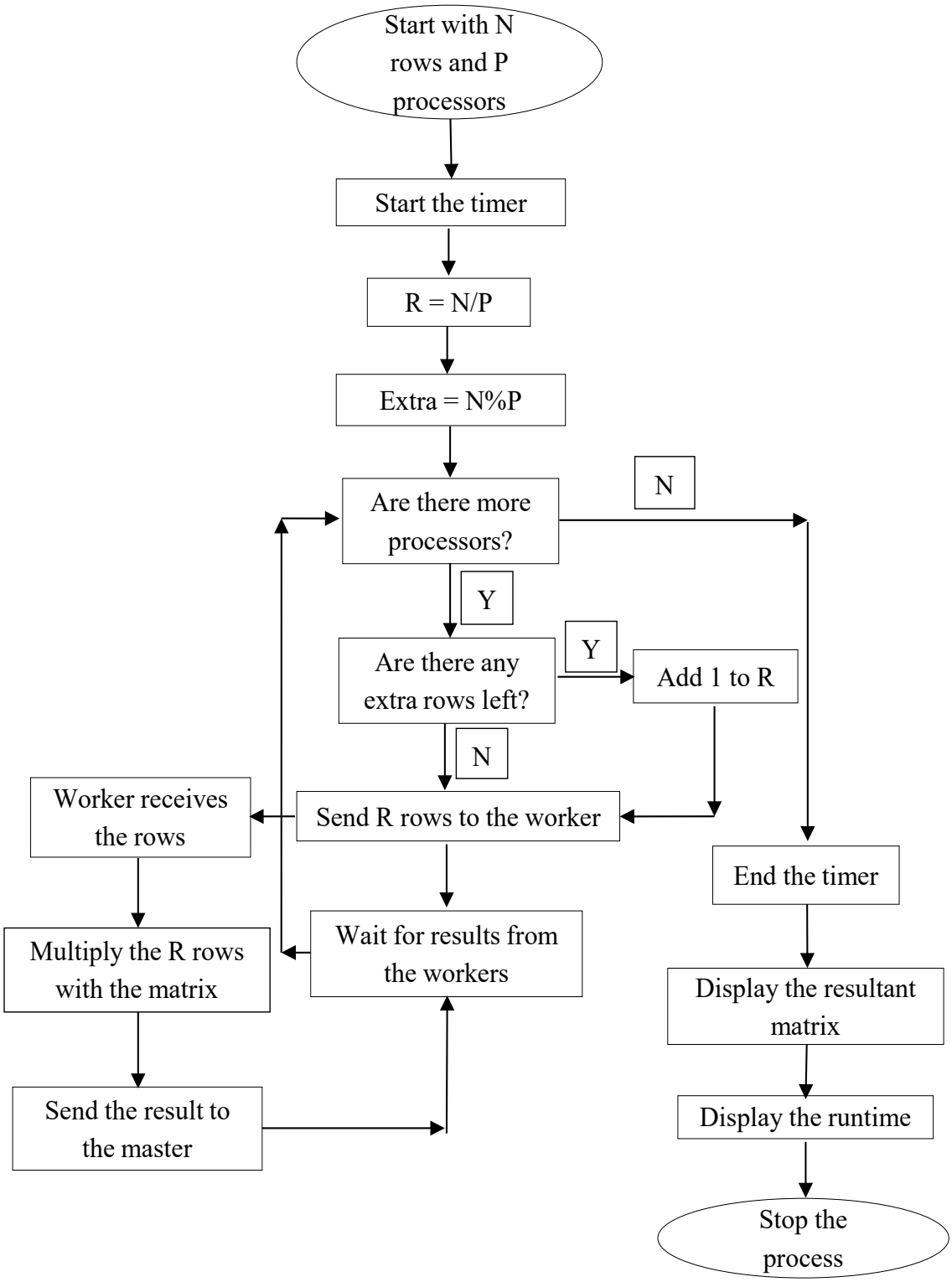


Figure 4. Flow chat for series to parallel conversion of 1D

5.0 Results and Discussion

5.1 Comparison of Computational Time with Number of Processors for 1D and 2D Problems

The variation of computational time with number of processors for 1D as well as 2D heat conduction is shown in the Figure 5. It observed that the computational time are decreases with increases the number of processors. Also, observed that the with increasing the number of processors the computational time is decreased significantly up to number of processor five there after its decreases almost constant because the FEM code for particular problem is solved 4 x 4 matrixes in both the cases for 1D and 2D. Further, it is observed that when the problem of heat conduction of 1D and 2D the computational time is always less for solving the problem of 1D at the same number of processors.

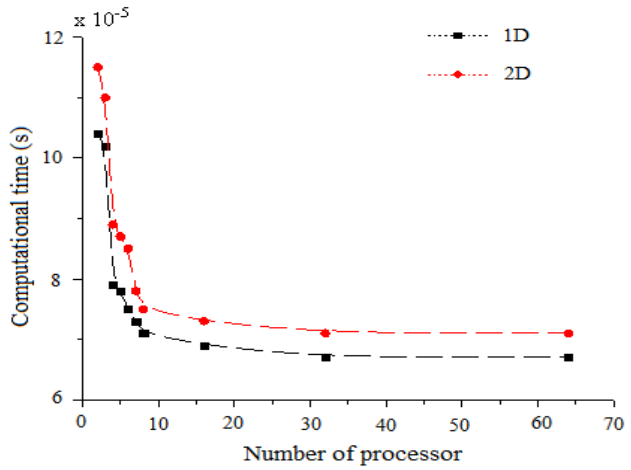


Figure 5. Variation of computational time with number of workers for 1D and 2D problems

5.2 Comparison of Speedup with Number of Processors for 1D and 2D Problems

The variation of speedup with number of processors for 1D as well as 2D heat conduction is shown in the Figure 6. The speedup is inversely proportional to computational time. It observed that the speedup is increases with the number of processors. The speedup is drastically increases between two to four processor after that the speedup is slightly increases almost constant because its solving the 4 x 4 matrixes. Further, it also observed that when the problem of related heat conduction with 1D and 2D the speedup is always high for solving the problem of 1D at the same number of processors.

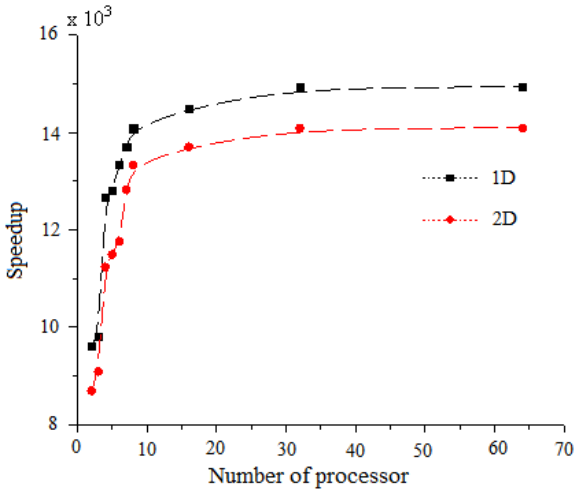


Figure 6. Variation of speedup with number of processors for 1D and 2D problems.

5.3 Comparison of Number of Operation (i.e., Iteration) with Number of Processors for 1D and 2D Problems

The variation of operation with increasing the processors for 1D and 2D problem is shown in the Figure 7. It is observed that the number of operation is varying linearly

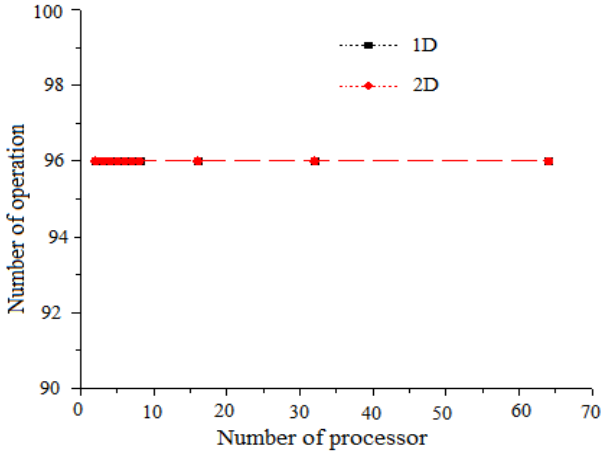


Figure 7. Variation of number of iteration with number of processors for 1D and 2D problems.

with the processors because it solves 4x4 matrixes in 1D and 2D heat conduction. Further, it is observed that the same number of operation (i.e., iterations) is taken for 1D as well as 2D problems.

5.4 Comparison of Speed in Meta-Flops with Number of Processors for 1D and 2D Problems

The meta-flops are a type of speed of computer which is directly associated with the variation of number of loops with number of processors for 1D and 2D problem is shown in Figure 8. It is observed that the number of meta-flops is proportional to speed up. The number of meta-flops is increases when the number of processors is increased. It is also observed that the number is meta-flops is more in 1D problem as comparative to 2D problem.

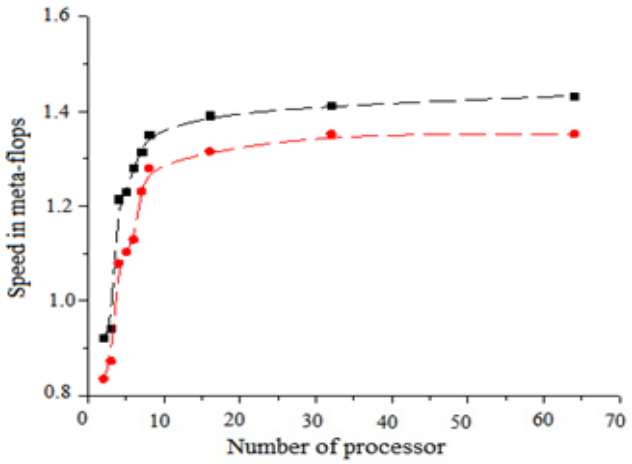


Figure 8. Variation of speed in meta-flops with number of processors for 1D and 2D problems.

6.0 GUI for the Developed Parallel FEM

Development of a GUI for parallel FEM 1D heat conduction problem aims to comfort use of the code while making it more users friendly and convenient to see and copy the results of the program. The designed a simplistic GUI which takes in the various parameters as input, as can be seen in the Figure 9. Similarly, develop the GUI for parallel FEM 2D heat conduction problem.

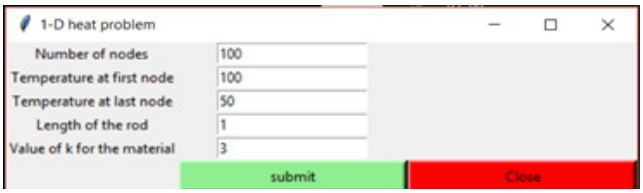


Figure 9. GUI for the developed parallel FEM for 1D heat conduction problem.

7.0 Conclusions

This paper dealt with the development of parallel codes of finite element analysis of 1D and 2D heat conduction problems encountered in mine ventilation networks. This work estimated the computational time, number of iteration, speedup, etc., using the parallel FEM codes at different processor with different size of the problems. The codes are developed in series as well as in parallel using MPI for finding the temperature at the unknown mesh of the nodes with varying temperature. The computational time decreased significantly up to fourth processor and then it decreased slowly even after increasing the processors. The speedup is found inversely proportional to the computational time. Further, it has been found that the number of iterations is same for 1D and 2D problem because the number of unknown nodes is equal. In sum, the parallel FEM can enhance the computational efficiency of large scale numerical problems.

8.0 References

1. Suri PK, Mitall S. A comparative study of various computing processing environments: A review. Int J Comp Sci Inf Technol. 2012; 3(5):5215-8.
2. Rodriguez G, Cannas ED. Parallel computing: A comparative analysis using Matlab [dissertation]. 2016.
3. Goyal A. Parallel computing using the multiscale finite element method for sub-surface flow models [dissertation]. 2016.
4. Babuška I, Elman HC. Some aspects of parallel implementation of the finite-element method on message passing architectures. J Computat Appl Math. 1989; 27(1-2):157-87. https://doi.org/10.1016/0377-0427(89)90365-8
5. Choporov SV. Parallel computing technologies in the finite element method. Radio Electron Compu Control. 2013; 2. https://doi.org/10.15588/1607-3274-2013-2-15

6. Wasy A, Iqbal M, Song J. Finite element simulation of simple bending problem and code development in C++. Eur Acad Res. 2013; 1(6):1391-405.
7. Riaz O. Parallel finite element methods and software for partial differential equations [dissertation]. University of Strathclyde; 2014.
8. Ahmad A. Parallel programming in the finite element methos. Proc Failure Eng Mat Struct. 2007; 87-93.
9. Rauber T, Rünger G. Parallel execution of embedded and iterated Runge-Kutta methods. Concurr Comput Pract Exp. 1999; 11(7):367-85. https://doi.org/10.1002/(SICI)1096-9128(199906)11:7<367::AID-CPE430>3.3.CO;2-7
10. de la Cruz LM, Monsivais D. Parallel numerical simulation of two-phase flow model in porous media using distributed and shared memory architectures. Geofís Int. 2014; 53(1):59-75. https://doi.org/10.1016/S0016-7169(14)71490-4
11. Wagner H. The management of heat flow in deep mines. Mining Report. 2013; 149(2):88-100. https://doi.org/10.1002/mire.201300014