ONE-DIMENSIONAL FINITE ELEMENT DISCRETIZATION OF CRACK PROPAGATION THROUGH PARALLEL COMPUTATION

Md. Rajibul Islam[1] and Norma Alias[2]
[1]Faculty of Information Science and Technology, Multimedia University, Malaysia
[2]Ibnu Sina Institute, Faculty of Science, University Technology Malaysia, Johor
E-mail: md.rajibul.islam05@mmu.edu.my, norma@ibnusina.utm.my

Abstract: *In this study, a new approach of the application of finite element method is presented, to solve the initial stages of crack propagation problems which mean the deformation due to the stress and strain of a material. In early applications of the finite element method for the analysis of crack propagation, the crack-tip motion was modelled by discontinuous jumps. We have implemented one dimensional finite element discretization to solve crack propagation problem. The parallel algorithm with parallel computer system has been used in order to perform the computational analysis of finite element for this study. Parallel Virtual Machine (PVM) has been used as a message passing software with Parallel Computer System. The result of this study will be useful in the mathematics and engineering fields. In mathematics, the research will widen the application of finite elements in solving the engineering science problems.*

Keywords: *Finite Element Method (FEM), Crack Propagation, Parallel computation, Parallel Virtual Machine (PVM).*

1 Introduction

Finite element method is a powerful technique that originally develops for structural analysis. Propagation problems refer to time-dependent, transient and unsteady-state phenomena. The method is applied to evaluate the stress intensity factors for plates of arbitrary shape using conventional finite elements [1].

PVM is a software package that permits heterogeneous collection of Linux environment as open space software hooked together by a network to be used as a single distributed parallel processor.

The most common applications are found in mechanics – solid mechanics, fluid mechanics, heat transfer and thermal stress analysis, couple problems, etc. A modern definition of the finite element method might state that it is simply a numerical procedure for finding approximate solution to boundary-value problems. In other words, it is to find a best-fit solution. Here, the value of the residual is minimized in some way to obtain the best-fit solution. In view of the fact that the method is approximation, so to archive such approximation there are four common methods to be used; collocation, subdomain integration, Galerkin, and least squares [2].

The basic concept of finite element method can be track through a series of papers which was published by Turner et al., Clough, Martin and Topp in 1956 [1, 3]. With these papers, the development of finite element in engineering applications began [3, 4]. The method was soon recognized as a general method of solution for partial differential equation.

We have divided this paper in the following way: In section 2, steps of the proposed finite element application is presented to solve one dimension crack propagation problem along with the C programming source code, parallel computation and performance measurement equations are explained in section 3, in section 4, a mathematical model of initial stages of crack propagation and discretization is constructed. Section 5 and 6 will describe numerical analysis and results and parallel performance estimation respectively and by the end of this paper, the conclusion has been presented.

2 Our Proposed Approach

We have implemented the following steps of finite element applications in order to solve the one dimension crack propagation problem,

Step (i): discretization of the domain,
Step (ii): selection of an interpolation or shape function,
Step (iii): derivation of element characteristic matrices and vectors,
Step (iv): assemblage of element characteristic matrices and vectors,
Step (v) solution of the system equations.

Below is the part of C programming source code that was developed to analyze one dimension crack propagation problem.

```
printf("Load Vector, F:\n");
fprintf(OutFile,"Load Vector, F:\n");
for(element=1;element<=e;element++)
{
        printf("f[%d]=\n",element);
        fprintf(OutFile,"f[%d]=\n",element);
        for(i=1;i<=e+1;i++)
        {
                if(i==element || i==element+1)

        f[element][i]=(Area[element]*l*0.2836)/2;
                else
                f[element][i]=0;
                printf("%15lf\n",f[element][i]);
                fprintf(OutFile,"%15lf\n",f[element][i]);
        }
        printf("\n");
        fprintf(OutFile,"\n");
}
printf("Global Load Vector, F:\n");
fprintf(OutFile,"Global Load Vector, F:\n");
for(i=1;i<=e+1;i++)
{
        for(element=1;element<=e;element++)
        {
                if(i==e+1 && element==e)
                GLV[i] = f[element][i]+F;
        }
}
for(i=1;i<=e+1;i++)
{
        printf("%20lf\n",GLV[i]);
        fprintf(OutFile,"%20lf\n",GLV[i]);
}
printf("\n");
fprintf(OutFile,"\n");
printf("Displacement, u:\n");
fprintf(OutFile,"Displacement, u:\n");
for(i=1;i<=e+1;i++)
        u[i][0]=0;
double j1=TOLERANCE,dif[30];
for(itr=1;(itr <= TIMESTEP)&&(j1>=TOLERANCE);itr++)
{
    for(element=1;element<=e+1;element++)
    {
    {
        for(j=1;j<=element-1;j++)
        sum1 += GSM[element][j]*u[j][itr];
        for(j=element+1;j<=e+1;j++)
        sum2 += GSM[element][j]*u[j][itr-1];
        u[element][itr] = (GLV[element]-sum1-
sum2)/GSM[element][element];
        }
        {
        for(j=1;j<=element-1;j++)
                sum1 += GSM[element][j]*u[j][itr];
        for(j=element+1;j<=e+1;j++)
                sum2 += GSM[element][j]*u[j][itr-1];
        u[element][itr] = (GLV[element]-sum1-
sum2)/GSM[element][element];
        }
        printf("u[%d][%d]=
%.15lf\n",element,itr,u[element][itr]);
        fprintf(OutFile,"u[%d][%d]=
%.15lf\n",element,itr,u[element][itr]);
        }
        printf("\n");
        fprintf(OutFile,"\n");
        j1=0.0;
        for(element=1;element<=e+1;element++)
        {
                dif[element]=fabs(u[element][itr]-
u[element][itr-1]);
                j1=(dif[element]>j1) ? dif[element] : j1;
        }
}
printf("Overall extension is %.15lf\n\n",u[element-1][itr-1]);
fprintf(OutFile,"Overall extension is %.15lf\n\n",u[element-
1][itr-1]);
for(element=1;element<=e;element++)
{
        Delu[element]=u[element+1][itr-1]-u[element][itr-
1];
        strain=Delu[element]/l;
        stress[element]=E*strain;
}
printf("The axial stress in each element is:\n");
fprintf(OutFile,"The axial stress in each element is:\n");
for(element=1;element<=e;element++)
{
    printf("stress[%d]= %15.15lf\n",element,stress[element]);
   fprintf(OutFile,"stress[%d]=
%15.15lf\n",element,stress[element]);
}
printf("\n");
fprintf(OutFile,"\n");
gettimeofday(&tv2,(struct timezone*)0);
dt1 = tv2.tv_sec-tv1.tv_sec;
dt2 = tv2.tv_usec-tv1.tv_usec;
if(dt2<0)
{
        dt--;
        dt2 = 1000000 +dt2;
}
printf("time=%d06%d\n"dt1,dt2);
fclose(OutFile);
```

3 Parallel Computation

Parallel Computing becomes an essential and vital problem solving standard for several computationally intensive applications, such as image processing, robotics, fracture mechanics [10]. The Parallel Virtual Machine (PVM) is a software tool for parallel networking of computers. It is designed to allow a network of heterogeneous machines to be used as a single distributed parallel processor. Such approach has proven to be a viable and cost-effective technology for parallel computing in many application domains [5]. The PVM system has gained widespread acceptance in the high-performance computing community.

High-performance computing (HPC) is a term that arose after the term "supercomputer". The term HPC refers to the use of parallel computers – that is computing systems comprised of multiple processors

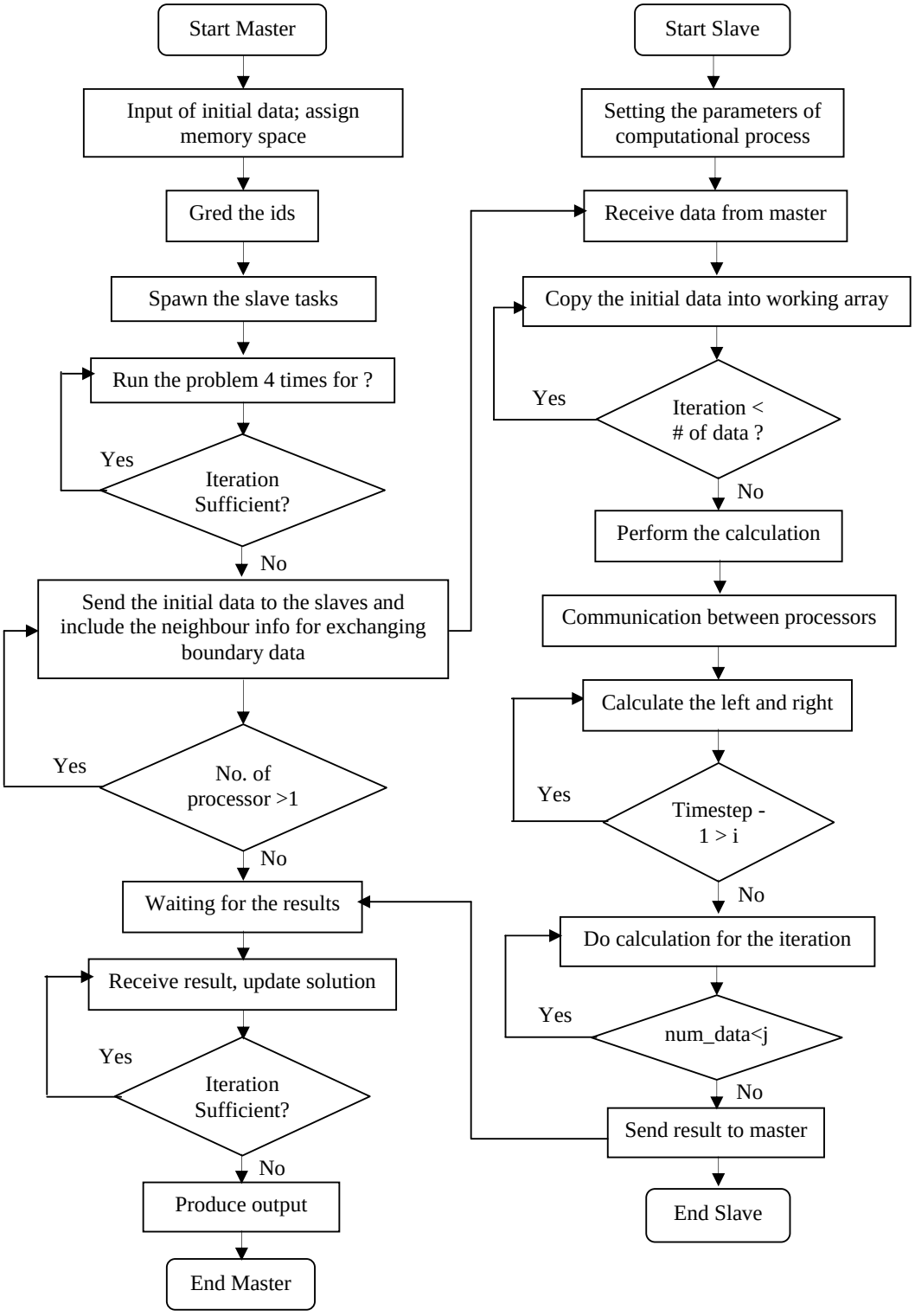

Fig. 1 Flow chart of parallel algorithm

linked together in a single system with commercially available interconnects [13]. The requirements of engineers and scientist for ever more powerful digital computers have been the main driving force in the development of digital computers [6].

Parallel Algorithm

According to [5], the performance of a parallel algorithm is assessed primarily by the following three factors:

i. Computing time (Time Complexity).
ii. Number of processors required (Processor Complexity).
iii. Model of the machine required.

There are some frequently used terms in parallel computing [5]:

1. Speedup: Wall-clock time of best serial execution divided by wall-clock time of parallel execution, also known as parallel speedup.

$$\text{Speedup} = \frac{T_s}{T_p} \qquad (1)$$

where

T_s = execution time for a single processor

T_p = execution time using $\boldsymbol{p}$ parallel processors

2. Efficiency: The efficiency is a measure of hardware utilization, equal to the ratio of speedup achieved on $\boldsymbol{p}$ processors to p itself.

$$\text{Efficiency} = \frac{Speedup}{p} \qquad (2)$$

3. Effectiveness: The effectiveness is used to calculate the speedup and the efficiency. It also can be said that the efficiency of a parallel program divided by the execution time.

$$\text{Effectiveness} = \frac{Speedup}{pT_p} \qquad (3)$$

4. Temporal performance: Temporal performance is a parameter to measure the performance of a parallel algorithm.

$$\text{Temporal} = \frac{1}{T_p} \qquad (4)$$

5. Scalability: A parallel system's ability to gain proportionate increase in parallel speedup with the addition of more processors.

4 The Discretization of Crack Propagation

Fracture mechanics is used to investigate the failure of brittle materials, which is to study material behaviour and design against brittle failure [7]. These failures arise as a consequence of unstable crack propagation from a pre-existing defect owing to material processing or fabrication [8].

The engineering study of fracture mechanics does not emphasize how a crack is initiated; the goal is to develop methods of predicting how a crack propagates, that is, how it lengthens [9]. The study of fracture therefore focuses primarily on the lengthening of a crack, and the resultant growth in surface area, as the load on the body is increased [12].

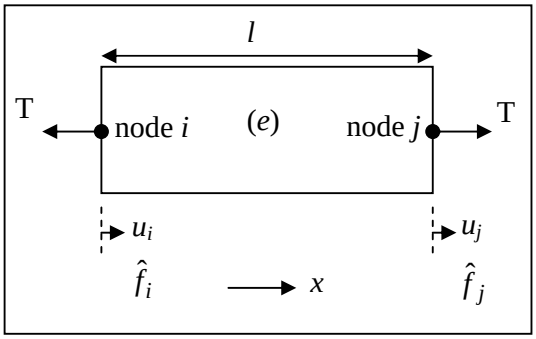

Fig. 2 One-Dimension Rod Element

Consider an element of length l of a bar subjected to an axial force as shown in figure 2. We know that the extension of the element is given by:

$$d\hat{u} = \frac{F\,dx}{AE} \Rightarrow \frac{d\hat{u}}{dx} = \frac{F}{AE} \qquad (5)$$

where $d\hat{u}$ is the extension of an element of length dx due to force, F. E and A are the Young's Modulus and constant cross sectional area of the element respectively. If F is constant over the element then $\frac{dF}{dx} = 0$.

Hence equation (5) becomes

$$AE\frac{d^2\hat{u}}{dx^2} = 0 \qquad (6)$$

Equation (6) is the governing equation for an axial element. Integrating over the length l, we get

$$AE\frac{d\hat{u}}{dx} = C_1 \qquad (7)$$

$$AE\,\hat{u} = C_1 x + C_2 \qquad (8)$$

Now, at $x = x_i, u = u_i$ and $x = x_j, u = u_j$, we have

$$AE\,u_i = C_1 x_i + C_2 \quad (9)$$

$$AE\,u_j = C_2 x_j + C_2 \quad (10)$$

from which

$$C_1 = \frac{AE(u_j - u_i)}{(x_j - x_i)} = \frac{AE}{l}(u_j - u_i)$$

$$C_2 = AE\,u_i$$

Therefore, $AE\,\hat{u} = \frac{AE}{l}(u_2 - u_1)x + AE\,u_i$

$$\text{or } \hat{u} = (u_j - u_i)\frac{x}{l} + u_i \quad (11)$$

After differentiation, we find from equation (11)

$$\frac{du}{dx} = \frac{(u_j - u_i)}{l} \quad (12)$$

At $x = x_i \rightarrow$

$$f_i = -AE\left(\frac{du}{dx}\right)_{x_i} = -AE\frac{(u_j - u_i)}{l}$$

At $x = x_j \rightarrow$

$$f_j = AE\left(\frac{du}{dx}\right)_{x_j} = AE\frac{(u_j - u_i)}{l}$$

The above can be expressed in matrix notation as

$$\begin{Bmatrix} f_i \\ f_j \end{Bmatrix} = \frac{AE}{l}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{Bmatrix} u_i \\ u_j \end{Bmatrix}$$

Finally, take

$$[K] = \frac{AE}{l}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Then

$$[K]\{u\} = \{F\} \quad (13)$$

where

$[K]$ = element stiffness matrix

$$= \sum_{e=1}^{E} k^{(e)}$$

$\{F\}$ = vector force

$$= \sum_{e=1}^{E} F^{(e)}$$

$\{u\}$ = constantly incremental displacement for border condition

To obtain $\{u\}$, we may use one of the numerical method, i.e. Gauss-Seidel Method [14].

$$u_i^{(l+1)} = \frac{f_i - \sum_{\substack{j=1 \\ i \neq j}}^{n} K_{ij} u_j^{(l+1)} - \sum_{j=i+1}^{n} K_{ij} u_j^{l}}{K_{ii}};$$

$$\forall\, i = 1, 2, \ldots n \quad (14)$$

After obtained the displacement, $u^{(e)}$ we can proceed to find the axial stress in each element, *e*.

$$\sigma_x^{(e)} = E\,\varepsilon_x = E\left(\frac{u_j - u_i}{l}\right) \quad (15)$$

The problem is solved by using the method discussed in section 2 and by using C programming and PVM as the platforms.

5 Numerical Analysis and Result

Based on the algorithm in Fig. 1, the solutions obtained are illustrated graphically with three different e to shown the stress-strain relationship between each node as well as each element.

According to Table 1, the relation between force applied to the element and stress generated can be concluded as: The axial stress is directly proportional to strain while strain is related to nodal displacement; such that:

and $\sigma_x \;\; \alpha \;\; \varepsilon_x$ communication and computational ratio $\varepsilon_x = \left(\frac{u_j - u_i}{l}\right)$

Table 1 Three different of discretization of the rod element for the analysis of one-dimensional continuum

Length (m)	Width (m)	Force (N)	No. of element (*e*)
10	2	100	5
10	2	100	10
10	2	100	15

Time execution for each different number of discretization of the domain increasing as the number of element discretises increase. Computational complexity is determined through the algorithm develop and shown as below:

Complexity = (number of operators used) ×

(number of iteration, k)
$= 31 \times k$
$= 31k$

Table 2 Time Execution for 1 CPU of three types number of element, *e*.

Number of Element, E	Time Execution for 1 CPU (sec.)
5	6.17E-03
10	6.32E-03
15	6.43E-03

6 Parallel Performance analysis

Based on the numerical results obtained, the performance measurements of parallel computing were analyzed from the aspect of time execution, speedup, efficiency, effectiveness and temporal performance. Fig. 2 is the time execution in second for 6 types of number of processors – 1, 4, 8, 12, 16 and 20. According to the graph, the time executes decreasing while the number of processors increases. This is because the task from the master had been divided into small parts to the slave. The more processors used means the more slaves the master can to divide the task. Thus as the number of processors increase, the time execute decrease.

Speedup: One way of judging the performance of an algorithm is to measure its speedup. This is because we are usually concerned to know about the performance gains from the algorithm over a similar algorithm run on a serial computer. Table 3 shows the speedup from the different number of processors.

Table 3 Time Execution and speedup of different number of processors

No. of Processors	Time Execution (sec.)	Speedup
1	60.660402	1
4	18.065915	3.357726525
8	10.629508	5.706793014
12	6.819884	8.894638384
16	6.524762	9.296952441
20	6.271058	9.673073029

From Fig. 4, we can see that as the number of processor increase, the speedup of the parallel algorithm also increasing. The results can be concluded as valid because in reality, when the more processors we used, the faster the calculation will performance.

Efficiency: The efficiency is used to judge how effective a parallel algorithm is. By formula (2), efficiency is measure through the fraction of time that a processor spends performing useful work. Table 4 shows the efficiency of the algorithm developed.

Table 4 Efficiency of the parallel algorithm

No. of Processors	Time Execution (sec.)	Efficiency
1	60.660402	1
4	18.065915	0.839431631
8	10.629508	0.713349127
12	6.819884	0.741219865
16	6.524762	0.581059528
20	6.271058	0.483653651

From the graph in Fig. 5 it shows that, while the number of processors increase, the efficiency of the parallel algorithm decreasing and all of them are less than 1 due to the communications involved within the processors. Following the equation (2), the speedup is increasing while the numbers of processors are also increase, thus the efficiency of the parallel algorithm decreasing.

Effectiveness: The efficiency of a parallel program divided by the execution time is known as the effectiveness of the parallel algorithm. Table 5 shows the effectiveness of the parallel algorithm developed.

Table 5 Effectiveness of the parallel algorithm.

No. of Processors	Time Execution (sec.)	Effectiveness
1	60.660402	0.016485219
4	18.065915	0.046464939
8	10.629508	0.067110268
12	6.819884	0.108685113
16	6.524762	0.089054517
20	6.271058	0.077124729

Fig. 6 shows the effectiveness of the parallel algorithm versus the number of processors. As the number of processors increase, the effectiveness are also increase but in this study, the effectiveness of the parallel algorithm decreasing when the number of processors exceeds twelve. This might be due to the communication problems within the processors.

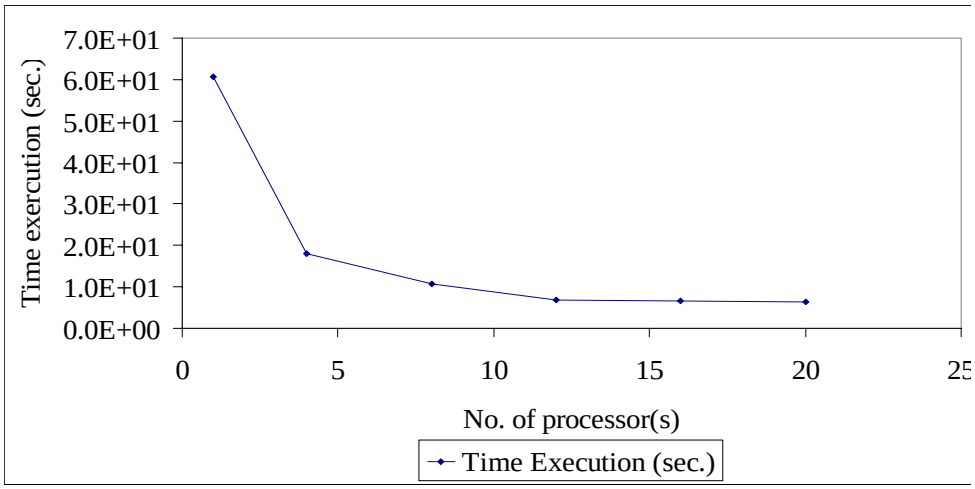

Fig. 3 Time Execution (sec.) for different number of processors.

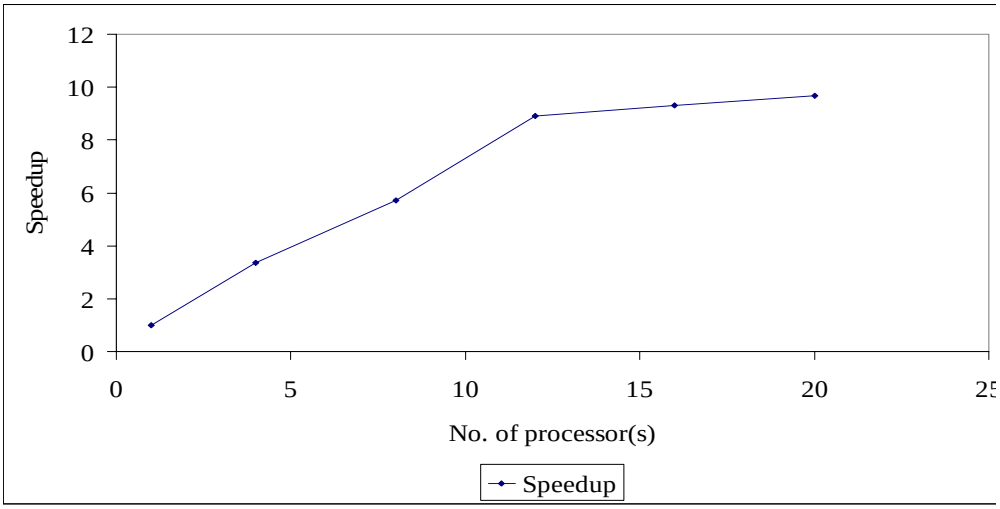

Fig. 4 Analysis of speedup for the different number of processors.

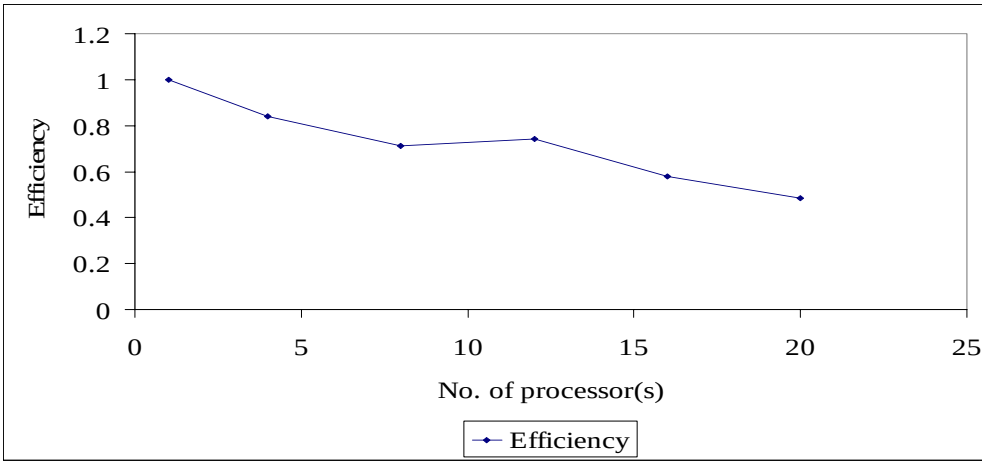

Fig. 5 Analysis of efficiency for the different number of processors.

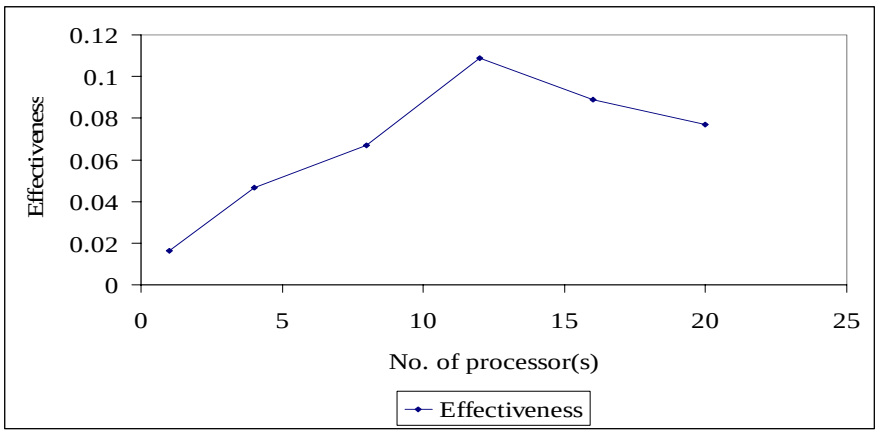

Fig. 6 Analysis of effectiveness for the different number of processors.

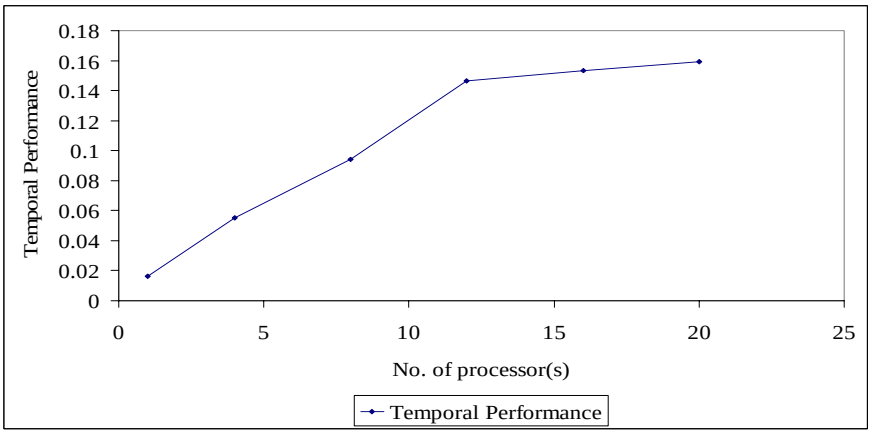

Fig. 7 Analysis of temporal performance for six different numbers of processors.

Temporal Performance: Temporal performance also used to analysis the performance of the parallel algorithm. Table 6 shows the temporal performance of the parallel algorithm developed in this research.

Table 6 Temporal performance of the parallel computer

No. of Processors	Time Execution (sec.)	Temporal Performance
1	60.660402	0.016485219
4	18.065915	0.055352856
8	10.629508	0.094077732
12	6.819884	0.14663006
16	6.524762	0.153262295
20	6.271058	0.159462725

Fig. 7 shows that, the temporal performance are increasing while the number of processors increases. This is because of the decreasing of the execution time as the number of processors used increasing.

After running the parallel computing based on 1, 4, 8, 12, 16 and 20 numbers of CPU, the parallel performance analyzed from the aspect of time execution, speedup, efficiency, effectiveness and temporal performance can be conclude as achieving the target of using parallel algorithm – to solve much larger problems at minimal time and by the same time increase the performance of the calculation.

The results have proven that parallel algorithm is better than the sequential algorithm or in other words, we can say that it is better than using a single processor. The computation of FEM is well suite in parallel algorithm because it involve in large scale of matrices and load vectors.

In this study, a C-programme as well as the PVM code has constructed to solve the problem. From the analysis of the performance of PVM, it has shown that the parallel computation using multi-processor is more efficient than the sequential computation using one processor in one PC while the parallel algorithm is used in order

to analyze the performance of the algorithm developed.

7 Conclusion

The finite element method is broadly used to analyze most engineering science problems [11]. In this study, we concern with the application of the finite element method to the calculation of the elastic stress and strain distribution in loaded bodies to solve the deformation problem. In this study, we have implemented the analysis of one-dimensional continuum in sequential algorithm using C-programming as well as the parallel algorithm using PVM. The solutions that will be obtained from parallel algorithm are expected to be more accurate and better than the result from sequential algorithm. Based on the numerical results and parallel performance evaluations, it's confirmed that parallel algorithm is an efficient solution of crack propagation prediction. The relationship between the applied force to the individual elements and the nodal displacement involved setting up the elements' stiffness matrices. Finite element methods are an alternative approach instead of finite difference discretization in terms of accurate prediction of crack propagation problem.

Acknowledgment

The authors acknowledge the Research Management Center, Institute of Ibnu Sina, UTM and Ministry of Science, Technology and Innovation Malaysia for the financial support (Grant no: 75019).

References

[1] Cheung, Y. K., Lo, S. H. and Leung, A. Y. T., Finite Element Implementation, Germany: Blackwell Science, Ltd. 1996.

[2] Chandrupatla, T. R. and Belegundu, A. D., Introduction To Finite Elements in Engineering, (2nd edition), New Jersey: Prentice Hall, Inc. 1997.

[3] Gutpa, K. K. and Meek, J. L., Finite Element Multidisciplinary Analysis. (2nd edition), Virginia: AIAA, Inc. 2003.

[4] Rao, S. S., Applied Numerical Methods for Engineers and Scientists, New Jersey: Pearson Education International, 2002.

[5] Xavier, C. and Iyengar, S. S., Introduction To Parallel Algorithms, Canada: John Wiley & Sons, Inc. 1998.

[6] Gray, J. P. and Naghdy, F., Parallel Computing: Technology and Practice, Australia: IOS Press. 1994.

[7] Bannantine, J.A., Comer, J.J. and Handrock, J.L., Fundamentals of Metal Fatigue Analysis, USA: Prentice-Hall, Inc. 1990.

[8] Henry, H., Levine, H., "Dynamic Instabilities of Fracture Under Biaxial Strain Using a Phase Field Model", Phys. Rev. Lett., Vol. 93, No. 10, 2004, pp. 105504.

[9] Bui, H. D., Fracture Mechanics: Inverse Problems and Solutions, The Netherlands: Springer, 2006.

[10] Alias, N., Sahimi, M.S., and Abdullah, A.R., "The AGEB Algorithm for Solving the Heat Equation in Two Space Dimensions and Its Parallelization on a Distributed Memory Machine", Proceedings of the 10th European PVM/ MPI User's Group Meeting: Recent Advances In Parallel Virtual Machine and Message Passing Interface, Vol. 7, 2003, pp. 214–221.

[11] Lewis, R. W., Nithiarasu, P. and Seetharamu, K. N., Fundamentals of the Finite Element Method for Heat and Fluid Flow, England: John Wiley & Sons Ltd., 2004.

[12] Stanley, P. ed., Fracture Mechanics in Engineering Practice, London: Applied Science Publishers Ltd., 1977.

[13] Wilkinson, B. and Allen, M., Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers, Prentice Hall, Upper Saddle River, New Jersey, 1999.

[14] Norma Alias, Md. Rajibul Islam and Nur Syazana Rosly, "A Dynamic PDE Solver for Breasts' Cancerous Cell Visualization on Distributed Parallel Computing Systems", in Proc. of *The 8th International Conference on Advances in Computer Science and Engineering (ACSE 2009)*, Phuket**,** Thailand, 2009, pp. 138-143.

Md. Rajibul Islam received his Bachelor of Computer Applications (BCA) degree from the Indira Gandhi National Open University, New Delhi, India, in 2004 and just completed his M.Sc degree in Information Technology from Multimedia University, Melaka, Malaysia. Currently he is working as a Research Assistant at Ibnu Sina Institute for Fundamental Science Studies, in Science Faculty of University Technology Malaysia, Johor. His research interests include High Performance Computing (HPC), Numerical Computation, Pattern Recognition, Image Processing, Computer Vision, and Artificial Intelligence.

Dr. Norma Alias obtained her PhD (Industrial Computing: Parallel Computing) from National University of Malaysia in 2004 and her M.Sc degree in Industrial Computing and BSc in Mathematics from the same University in 1997 and 1991 respectively. Currently, she is working as a Senior lecturer at the Mathematics Department in Faculty of Science, University Technology Malaysia (UTM) and the Researcher Head of High Performance Computing Group as well in Ibnu Sina Institute for Fundamental Science Studies, UTM. She has published over 100 papers in several referred International journals, conferences, workshops, lecture notes and book chapters. She is a reviewer of several International Journals with high impact factor and association with some International conferences and workshops organized by Ibnu Sina Institute. Her research interests include Industrial Computing, Numerical Computation, and Scientific Computing & High Performance Computing on Distributed Parallel Computer Systems.