Applying Parallel Programming to the *N Scheme* for solving FEM cases without assembling an $A\,x = b$ system

J. Eyng[1], J.P.A. Bastos[1], N. Sadowski[1], M. Fischborn[2], M.A.R. Dantas[3]
[1]GRUCAD/EEL/CTC, Universidade Federal de Santa Catarina, CP. 476, Florianópolis, SC, 8804-900, Brazil
[2]UTFPR, Universidade Tecnológica Federal do Paraná, Campus Medianeira
[3] LaPeSD/INE/CTC, Universidade Federal de Santa Catarina, CP. 476, Florianópolis, SC, 8804-900, Brazil
eyng@grucad.ufsc.br, jpab@grucad.ufsc.br, nelson@grucad.ufsc.br,
fisch@utfpr.edu.br, Mario@inf.ufsc.br

Abstract — In previous works we introduced the *N Scheme* method, which is a much simpler technique to solve FE cases compared with classical methods [1]. Traditional methods require assembling, storing and solving an $Ax = b$ matrix system. With this technique the assembling of the matrix A is unnecessary. The drawback of the proposed method is the computational time: it requires more iterations to converge, even though the results are reliable. One possible way to improve its performance is the application of parallelization techniques as firstly introduced in [2].

I. INTRODUCTION

Since this technique is still relatively new, to improve the understanding of this paper, we replicate here some parts of [1] and [2] where the *N Scheme* method was introduced. The new technique, calculates the unknown potential nodes of the mesh in a way much simpler than the traditional ones used for solving FE cases. In conjunction with the techniques of parallel programming, the proposed work provides correct results and the processing time for solving large problems, typically 3D problems, is well reduced.

II. THE METHOD

The *N Scheme* works by nodes (instead the classical assembling by elements) using cells of elements around nodes i.e., the elements having the vertex node *n*. The method is solved as follows:

- For each *n* unknown node:
  - ✓ Calculate the elemental matrices of all the elements of the cell around *n*, making the appropriate sums and operating as shown in [1];
  - ✓ Thus, we have a first approximation of the potential for this node unknown;
- Repeat the process with the new magnetic potential calculated on the nodes, until the convergence is obtained.

The operations are identical to the Gauss-Seidel method and for reaching convergence, it is necessary to apply a relaxation factor to the solution. It is also known as SOR (Successive Over Relaxation method) [1].

III. IMPLEMENTATION

The parallelization technique is applied after the division of the finite element mesh. This division is made so that we can distribute parts of the mesh among various processes. Each process uses one different parallel processor in the cluster to calculate simultaneously the electric potential on nodes using the *N Scheme* method.

The processes receive their part of the mesh and execute the *N Scheme* method. Each process returns its results to the master process. This procedure is performed until convergence is achieved.

The objective of this work is to show that the *N Scheme* method can deliver results with the application of parallelization techniques. It compares parallel solutions with known solutions of the FE method. Other methods such as the application of Conjugate Gradient to the *N Scheme*, are also under consideration now for improving the performance of the method.

The parallel program development is processed here using MPI library [3][4]. The environment used for testing were *Mirynet* clusters, available in *Sharcnet*[1] (Shared Hierarchical Academic Research Computing Network).

IV. THE PROBLEM

The geometry used for the simulations is a cube composed of a single material with Dirichlet imposed boundary condition on nodes of the upper and lower surface geometry. We have worked with a mesh of hexahedral elements generated in the program. Thus, we developed a program to solve an electrostatic problem, calculating the potential on the unknown nodes of the mesh.

The division of the problem is made according to the geometry as show in Fig.1; we used a fixed number of divisions equal to six for different mesh sizes.

The *N Scheme* method needs the elemental matrices of the hexahedra. For speeding up the calculation it is required that such matrices are stored in the memory. However, at this stage of our work, we opted for a regular FE mesh, the elemental matrices are identical for all elements. Therefore the storing is minimal.

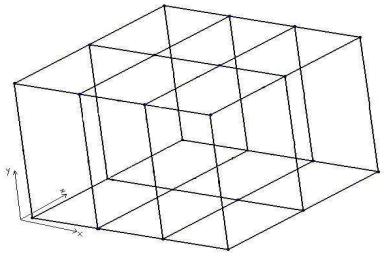

Fig. 1. Division of the geometry

[1] SHARCNET homepage, http:/www.SHARCNet.ca

V. SIMULATION

The program was development using Fortran 77 in conjunction with the MPI communication library. The simulation results were obtained working in clusters and eight processors for the simulations were provided by *Sharcnet*.

One of these processors is the master, responsible for managing the distribution of tasks among the processes and awaiting the results by processors. In this implementation, the master process does not perform the calculation of the potential on the nodes of the mesh.

The objective of this proposal is to achieve good results in parallel processing time compared with the results of sequential program time. The sequential and parallel programs were running in a cluster operating with the system HP Linux 3.1.

The FE meshes reached millions of elements in order to depict good performances on the parallel program compared to the sequential program. For all simulations, we used a precision of 10^{-5} with the final relaxation factor of 0.96 and time was measured in seconds.

VI. RESULTS

The results of processing time shown in Table I and Fig.2 were obtained in the *Myrinet* cluster for 3D meshes.

TABLE I
PROCESSING TIMES AND NUMBER OF ITERATIONS IN SEQUENTIAL AND PARALLEL PROGRAMS

Number of nodes	Time in sequential	Time in parallel	Number of iterations in sequential	Number of iterations in parallel
157464	6.81	13.87	169	250
373248	18.26	30.39	186	249
884736	55.12	80.94	209	275
2000376	164.7	195.94	275	291
2299968	206.61	239.68	302	306
3796416	485.3	517.88	408	369
5268024	785.03	838.96	489	415
7077888	1921.53	1234.05	572	449
8489664	1670.62	1455.5	628	460
10941048	2816.21	1953.41	715	473

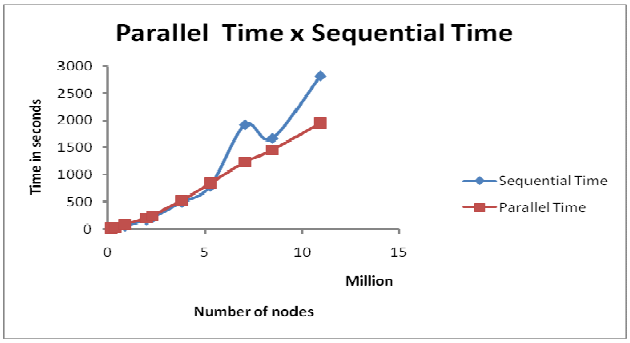

Fig. 2. Processing time in sequential and parallel programs

Table II shows de behavior of values for the speedup and efficiency for simulations.

TABLE II
VALUES OF THE SPEEDUP AND EFFICIENCY

Number of nodes	Speedup	Efficiency
157464	0.53835	0.08973
373248	0.63104	0.10517
884736	0.72357	0.12060
2000376	0.81376	0.13563
2299968	0.86923	0.14487
3796416	1.02378	0.17063
5268024	1.06664	0.17777
7077888	1.62517	0.27086
8489664	1.30332	0.21722
10941048	1.28195	0.21366

According to the results, the processing time of parallel program becomes interesting when mesh size increases. Therefore, the FE mesh needs to reach a considerable size (over 700,000 unknowns) to justify the use of parallel programming.

VII. CONSIDERATIONS

This work is still in progress. The *N Scheme* method has a processing time larger when compared with a classical FEM implementation using, for instance, an ICCG solver. In order to improve the *N Scheme* we proceeded with investigations on the Conjugate Gradient (CG) method, while keeping the main principles of the *N Scheme* method, i.e., not assembling the matrix *A* [5]. Our proposal is to apply parallel programming on the way here described to parts of CG algorithm which seem to be appropriate. The corresponding results will be presented in the full paper.

VIII. REFERENCE

[1] J.P.A. Bastos, “Is it possible to solve a FEM static case without assembling, storing and solving an Ax = b matrix system?” *ICS Newsletter*, vol 16, number 1, March 2009, UK

[2] Eyng, J.; Bastos, J.P.A.; Sadowski, N.; Fischborn, M.; Dantas, M.A.R.; Ferreira, D.J., “Parallel programming apllied to the N Scheme for solving FE cases without assembling an Ax = b system.” *The 14th Biennial IEEE Conference on Electromagnetic Field Computation – CEFC 2010*, vol. 1, p. 1, May de 2010, Chicago, Illinois, USA

[3] J. Dongarra et al, “*MPI: the complete reference*”. Massachusetts Institute of Tecnology, (http://www.netlib.org/utk/papers/mpi-book/mpi-book.html)

[4] H. Jordan, G. Alaghband, *Fundamentals of Parallel Processing*, Prentice-Hall (Upper Saddle River, N.J), 2003.

[5] R. Barret, M. Berry, T.F. Chan, J. Demmel, J.M. Donatto, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. Van der Vorts, “*Templates for the solution of linear systems: building blocks for iterative procedures*”, Society for Industrial and Applied Mathematics, (http://www.siam.org/books)