Parallel Implementations of the PEEC Method

Danesh Daroui and Jonas Ekman

Department of Computer Science and Electrical Engineering
Luleå University of Technology, 971 87 Luleå, Sweden
danesh.daroui@ltu.se, jonas.ekman@ltu.se

Abstract **─** This paper presents the first parallel implementation of a partial element equivalent circuit (PEEC) based electromagnetic modelling code suitable for solving general electromagnetic problems. The parallelization is based on the GMM++ and ScaLAPACK packages which are cross-platform libraries available for major operating systems. The parallel PEEC solver has been tested on several high performance computer systems. Large structures containing over 250 000 unknown current and voltage basis functions were successfully analyzed for the first time with a general PEEC-solver. The numerical examples are of orthogonal type, studied both in the time and frequency domain, for which memory, performance, and speed-up results are presented.

Index Terms **─** PEEC, parallel computing, integral equation.

I. INTRODUCTION

As for all the methods within computational electromagnetics, the problem system size that can be solved increases with more efficient computer implementations and more powerful computer systems. However, the desired problem sizes to be solved also increase and there is a clear gap between desired and possible problem size to be solved. Fast solutions for EM problems have been treated for a long time, i.e. [1] where both differential and integral equation solvers were discussed. For the integral equation based solvers, fast Krylov subspace approaches are available, for example, the fast-multipole method (FMM) [2] and QR-based algorithms [3]. The next step, after faster implementations, is to improve the computing power running the algorithms. One solution is to use grid computing on different levels. For example, using a local area network of interconnected computers to speed-up calculations or by porting the code to parallel architectures. Recent publications on the extension to parallel implementations are for example [4] where a nesting combination of the finite element domain decomposition method and the algebraic multigrid method is presented, [5] on the implicit FDTD method, and [6] for a parallel version of the numerical electromagnetics code (NEC).

The partial element equivalent circuit (PEEC) method [7] is widely used for solving mixed circuit and electromagnetic (EM) problems. The method gives a framework for creating electric equivalent circuit representations for three-dimensional electromagnetic problems and calculating self and mutual partial inductances [8] and capacitances (coefficients of potential) [9]. The resulting equivalent circuits can be solved in SPICE-like solvers or, for the full-wave case, by creating and solving the fully coupled circuit equations [10]. Until now, no parallel implementation on the PEEC method has been reported except for in [11] where a sequential code was parallelized for LANs using a freeware. In this paper, the first parallel implementation of a non-accelerated, e.g. FMM, PEEC method [12] is presented for high performance computing using the ScaLAPACK package [13].

Other approaches for accelerating PEEC-based computations are for example FMM-based approaches as detailed in [14, 15], wavelet-based PEEC analysis as in [16-18], and QR-decomposition as shown in [19]. The goal with this work has been to accelerate the general PEEC method which allows for both time and frequency domain solution from DC to the highest frequency of interest (given by the mesh) and not to be restricted by the above mentioned acceleration techniques impacting, for example, on the low frequency behaviour. The paper is organized in the following way. Section II presents a summary of the PEEC method and the developed computer program while Section III presents the parallelization of the same using the ScaLAPACK package. Then, Section IV and V show the applicability of the solver for two numerical examples, a free-space reactor and a shielding study. Finally conclusions and further work are detailed in Section VI. It is shown that with this type of parallel PEEC solvers the problem size can be increased considerably and new application areas arise.

II. SUMMARY OF PEEC THEORY

This section gives a brief summary of the classical, orthogonal PEEC formulation. For further information, see [7-9].

A. Extraction of Equivalent Circuit

The classical PEEC method is derived from the equation for the total electric field at a point [20] written as

$$\boldsymbol{E}^{i}(\boldsymbol{r},t)=\frac{\boldsymbol{J}(\boldsymbol{r},t)}{\sigma}+\frac{\partial \boldsymbol{A}(\boldsymbol{r},t)}{\partial t}+\nabla\varphi(\boldsymbol{r},t), \qquad (1)$$

where E^{i} is an incident electric field, J is a current density, A is the magnetic vector potential, ϕ is the scalar electric potential, and σ the electrical conductivity all at observation point $\boldsymbol{r}$. By using the definitions of the scalar and vector potentials, the current- and charge-densities are discretized by defining pulse basis functions for the conductors and dielectric materials. Pulse functions are also used for the weighting functions resulting in a Galerkin type solution. By defining a suitable inner product, a weighted volume integral over the cells, the field equation (1) can be interpreted as Kirchhoff's voltage law over a PEEC cell consisting of partial self inductances between the nodes and partial mutual inductances representing the magnetic field coupling in the equivalent circuit. The partial inductances shown as L_{p11} and L_{p22} in Fig. 1 are defined as

$$L_{p\alpha\beta}=\frac{\mu}{4\pi}\frac{1}{a_{\alpha}a_{\beta}}\int_{v_{\alpha}}\int_{v_{\beta}}\frac{1}{\left|r_{\alpha}-r_{\beta}\right|}dv_{\alpha}dv_{\beta}\ , \qquad (2)$$

for volume cell α and β. Figure 1 also shows the node capacitances which are related to the coefficients of potential p_{ii} while ratios consisting of p_{ij}/p_{ii} are leading to the current sources in the PEEC circuit. The coefficients of potentials are computed as

$$p_{ij}=\frac{1}{S_{i}S_{j}}\frac{1}{4\pi\varepsilon_{0}}\int_{S_{i}}\int_{S_{j}}\frac{1}{\left|r_{i}-r_{j}\right|}dS_{j}dS_{i}\ \ , \qquad (3)$$

and a resistive term between the nodes, defined as

$$R_{\gamma}=\frac{l_{\gamma}}{a_{\gamma}\sigma_{\gamma}}\ . \qquad (4)$$

In (2) and (4), *a* represents the cross section of the rectangular volume cell normal to the current direction γ, and l is the length in the current direction. Further, *v* represents the current volume cells and *S* the charge surface cells. For a detailed derivation of the method, including the nonorthogonal formulation, see [21].

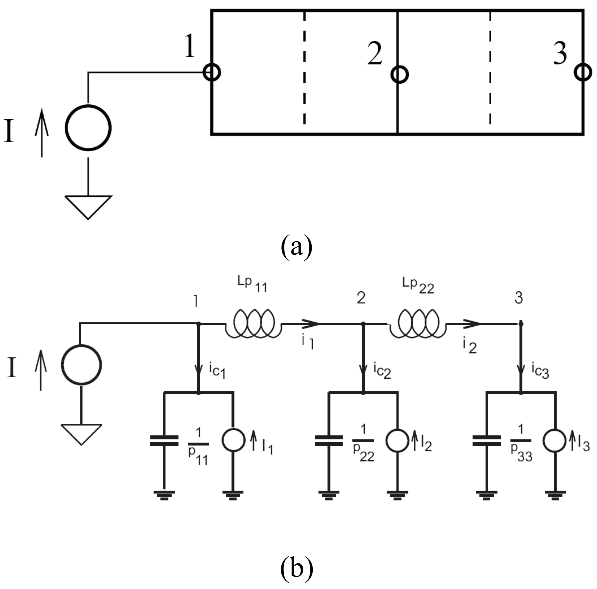


Fig. 1. Metal strip with 3 nodes and 2 cells (a) and corresponding PEEC circuit (b).

B. Solution of Equation of Circuit

The discretization process of the EFIE in (1) and the successive Galerkin's weighting leads to an equivalent circuit formulation. When Kirchhoff's voltage and current laws are enforced to the N_i independent loops and N_ϕ independent nodes of the PEEC equivalent circuit we obtain:

$$\begin{aligned} &-A\Phi(t) - Ri_L(t) - L_p \dot{i}_L(t) = v_s(t), \\ &P^{-1}\Phi(t) - A^T i_L(t) = i_s(t), \end{aligned} \tag{5}$$

where

- $\Phi(t) \in \Re^{N_\phi}$ is the vector of node potentials to infinity; $\Re^{N_\phi}$ is the node space of the equivalent network;
- $i_L(t) \in \Re^{N_i}$ is the vector of currents including both conduction and displacement currents; $\Re^{N_i}$ is the current space of the equivalent network;
- L_P is the matrix of partial inductances describing the magnetic field coupling;
- P is the matrix of coefficients of potential describing the electric field couplings;
- R is the matrix of resistances;
- A is the connectivity matrix;
- $v_s(t)$ is the vector of distributed voltage sources due to external electromagnetic fields or lumped voltage sources;
- $i_s(t)$ is the vector of lumped current sources.

The equation system in (5) is equivalent to the circuit equations formulated in SPICE-type of solvers for obtaining the solution in node voltages and branch currents. However, for PEECs the equation system in (5) contain more dense matrices (L_P and P) compared to a pure electric network system solution due to the large number of mutually coupled inductors and mutual capacitances. Therefore, the solution of PEECs requires linear algebra packages suitable for dense matrices. The exception is the full-wave, time domain case where retarded magnetic and electric field couplings are treated as known sources and the L_p and P matrices are more sparse [10].

The equation system in (5) is often entitled a Modified Nodal Analysis (MNA) formulation [22] and can be modified to suit the solution of PEECs [10]. From the MNA formulation, the Nodal Analysis (NA) formulation can be derived which only solves for the node potentials by a reduced equation system while the branch currents are calculated in a second step. In the frequency domain the NA system can be written as

$$\Phi(\omega) = \left[-A^T (R + j\omega L_p(\omega))^{-1} A + j\omega P(\omega)^{-1}\right]^{-1} I_S . \tag{6}$$

to solve for the node potentials Φ at a specific frequency for the excitation specified by I_S. Both formulations are tested in this paper and results are presented in Sec. IV and V.

C. Sequential Code for EM Analysis Using PEEC Theory

A program for EM analysis, based on the theory and references outlined above, has been developed [23]. The solver can handle both the traditional orthogonal PEEC model and the newly introduced nonorthogonal formulation [21]. In this paper, only orthogonal models are considered while nonorthogonal results will be presented in a future paper since different issues arise when working with nonorthogonal PEEC models [24], [25]. The program creates an equivalent circuit and calculates the corresponding resistances, partial inductances, capacitances, and coupled voltage and current sources (to account for electromagnetic couplings) for the given geometrical layout (CAD-data as specified in an input file). The user adds external electronic (sub-) systems and analysis mode as described by the SPICE syntax. The actual solution of the resulting circuit equations (5) in either the time or frequency domain is performed in the solver and results are given as current- and voltage distributions in the geometrical layout. Post-processing routines are implemented for calculating field quantities at specified locations. The workflow in the program is shown in Fig. 2.

The sequential implementation utilizes the GMM++ linear algebra package and the Intel C++ Compiler with pragmas for compiler optimization to be performed. This allows, for example, for the use of multiple processors in calculating partial elements and other trivial pipelining, loop unrolling/distribution, data prefetching, and loop-carried dependencies occurring in the original, sequential implementation.

It is the presented sequential code that has been parallelized and for which results are presented in this paper.

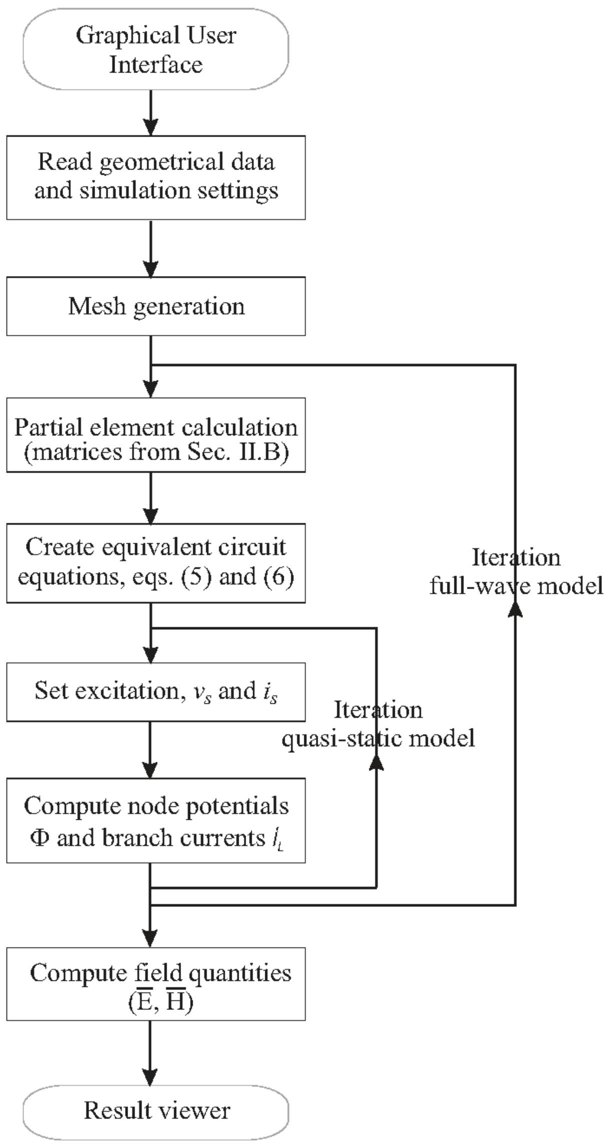

Graphical User Interface
Read geometrical data and simulation settings
Mesh generation
Partial element calculation (matrices from Sec. II.B)
Create equivalent circuit equations, eqs. (5) and (6)
Iteration full-wave model
Set excitation, v_S and i_S
Iteration quasi-static model
Compute node potentials Φ and branch currents i_L
Compute field quantities ($\overline{E}$, $\overline{H}$)
Result viewer

Fig. 2. Flow diagram for the PEEC solver.

III. PARALLELIZATION OF THE PEEC SOLVER

A. Introduction

The development platform was a Linux cluster consists of nodes equipped with two Intel Xeon quad-core 2.5 GHz CPUs and 16 GB of RAM memory. The code has been written in C++ under Linux and is compatible with parallel computer systems with distributed memory architecture using ScaLAPACK as computational library. ScaLAPACK (Scalable LAPACK) is a library of LAPACK routines, revised for parallel computer systems with distributed memory architecture. The package enables the use of high performance computing clusters in a simple fashion and allows for a considerable acceleration of the developed PEEC-based program. Like LAPACK, ScaLAPACK offers a set of highly optimized routines to solve systems of linear equations, which consists of matrices distributed among a bunch of processors. The library performs basic linear algebra operations such as product between matrices and vectors using PBLAS. PBLAS is parallel version of a rich library of computational routines called BLAS which is included in LAPACK. Finally a set of routines called BLACS is used to manage communication between nodes running ScaLAPACK. These routines use algorithms called block-partitioned algorithms to minimize movement of data between nodes by load balancing between computational elements. ScaLAPACK has been written in FORTRAN and developed for parallel computer systems. The choice to use this library was based on optimized and efficient message passing methods have been used in it, speed and scalability and good interface for C++ programmers. In addition it is a stable, well tested, and efficient library and provides access to a very large collection of useful, powerful and flexible functions in BLAS and LAPACK which have been parallelized efficiently. Using ScaLAPACK we were assured that a good load balancing is achieved by distributing input data on a bunch of processing nodes using block cyclic data distribution algorithms which speeds up the operations by minimizing data transfer between processing units.

The parallel solver performs these four steps to solve a problem:

1. The discretization process is entirely serial and duplicated on all processors.
2. The partial element calculations are easily parallelized as no communication is required between nodes while each node calculates assigned part of basic matrices in parallel with other nodes. The main difficulty lies in the mapping between global and local matrix coordinates [6].

3. The matrix formulation, (5) or (6), and solution parts are implemented using ScaLAPACK routines.
4. At the end when all processes have reached final synchronization point, the results will be gathered on the root processing unit and will be saved in appropriate format.

B. Parallelization of Partial Element Computations

The partial element calculations are easily parallelized using parallel processors which fill a large matrix, distributed by ScaLAPACK data management algorithms, completely in parallel and independent of each other. For the time domain problems these are L_P and P matrices which are symmetric and have entries of type double precision floating-point. Hence the fill-in times for the time domain solver is decreased linearly as number of allocated processors grows. Since these two matrices are symmetric, Cholesky factorization routines in ScaLAPACK package can be used to factorize them. Due to the properties of the entries of L_P and P matrices which has the type complex with double precision floating point for problems in frequency domain and because these matrices do not fulfill Hermitian properties, only LU factorization is possible and therefore the need to fill in the whole matrix. The process of placing element from one part of a distributed matrix to the other part is computationally expensive and complicated in parallel programs and especially for the MNA-approach seen in (5). But this was overcome by a special Transpose-And-Add method as detailed in [26].

C. Parallelization of Matrix Solutions

After filling-in the matrices, the solution of the time or frequency domain versions of the circuit equations in (5) and (6) has to be performed. This is done using the ScaLAPACK library of high-performance linear algebra routines for distributed memory message-passing MIMD (multiple instruction stream, multiple data stream) computers and networks of workstations supporting parallel virtual machine (PVM) and/or message passing interface (MPI). ScaLAPACK uses block cyclic data distribution [27] to achieving good load balancing. This means that matrices are divided into blocks in two dimensions and these blocks are assigned to a set of processors. This is further detailed in [6] when using the numerical electromagnetic code (NEC) to solve electromagnetic problems using ScaLAPACK.

IV. NUMERICAL TEST (I) - AIR-CORE REACTOR

To present the speed-up of the parallel PEEC implementation, an air-core reactor structure is utilized since measurement results have been collected and the cell count is easily increased. In previous papers, i.e. [28], the reactor have been studied in the time and frequency domain with regular (L_P, R, P, τ) PEEC models. However, the inclusion of Skin and proximity effects have not been possible in earlier works through the volume filament approach, (VFI)PEEC, due to the excessive number of unknowns. Here, the air-core reactor is analyzed with both the original serial PEEC solver, when possible and the new parallel PEEC solver in the time and frequency domain.

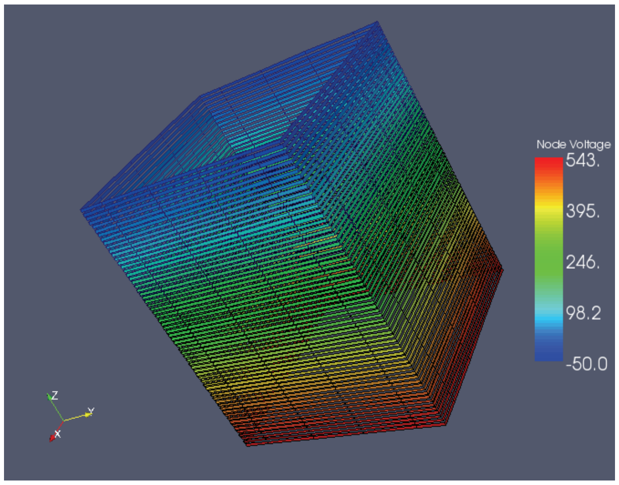


Fig. 3. Reactor voltage simulation result at 4 µs after impulse test.

The test structure, seen in Fig. 3, is of rectangular type with four sides equal in length = 0.5 m. The windings (turns) are totally 65 and consist of copper tape with dimension 0.076 mm x 6.35 mm. The center to center spacing between the turns is 10 mm. The parallel implementation of the solver can now treat this type of problem and give a more correct model for the current distribution in the conductors. Figure 3 shows an example of the voltage distribution in the reactor windings for a, time domain, impulse test.

A. Partial Element Calculation Speed-Up

To test the speed-up, five different meshes for the reactor are utilized as seen in Table 1. For example, the first test, T1-400, has 1300 surface cells, 1040 volume cells, resulting in 2340 unknowns, and 1 105 nodes. The last test, T5-609, has 50180 unknowns since it uses the volume filament approach to model Skin effect in the windings. The naming conventions used for these test cases is in the format of T[*n*]-[*abc*], where n is the test case number and *a*, *b* and *c* represent the discretization level in the directions *x*, *y* and *z* respectively.

The partial element calculations are efficiently parallelized as seen in Table 2. The table is collected from time domain simulations. To be noted is that the test case is an orthogonal PEEC model utilizing analytical routines to evaluate partial inductances (< 5μs/element in sequential solver). The performance gain for the parallel implementation, as shown in Table 2, can be displayed by using a speed-up factor

$$S(n) = \frac{t_{p_1}}{t_{p_n}} \quad , \tag{7}$$

where t_{p_1} is the time taken by the parallel code using one processor and t_{p_n} is the time taken by the parallel code using n processors. This is shown in Fig. 4 where results from Table 2 are used together with results from the frequency domain solver by running the same test cases in the frequency domain.

From the figure it is clear that the time domain fill-in time is better than the frequency domain fill-in. This is because the time domain solvers use symmetric matrices with the data type of double precision floating point, so they can use symmetric compatible functions in ScaLAPACK which employ Cholesky factorization. But in the frequency domain, since non-Hermitian matrices with the data type of complex with double precision floating point are used, ScaLAPACK does not offer any symmetric compatible function. Therefore for frequency domain solvers the calculated part of matrix needs to be copied to the other part to form the complete matrix and this process will affect the speed-up factor as is shown in Fig. 4.

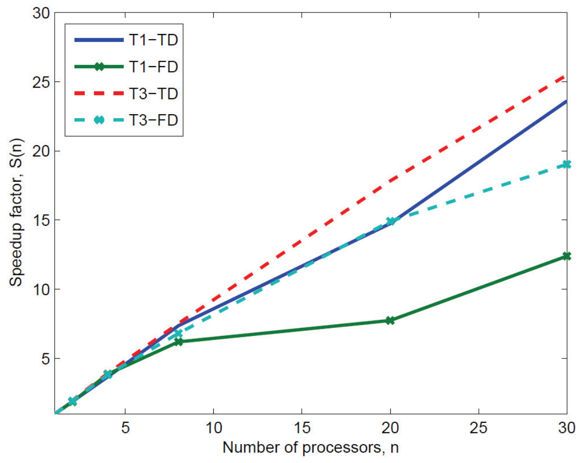


Fig. 4. Speed-up factor for partial element calculation and fill-in for air-core reactor example.

B. Solution Method - MNA or NA

From the calculated matrices, as briefly detailed in Sec. II-B, two popular methods are used to formulate the circuit equations for the PEEC model. First, the MNA formulation as shown in (5) and second the NA formulation as shown in (6). The MNA formulation is the more general of the two and preferred mainly due to its ability to handle general circuit element inclusion with the PEEC model [22] and a stable low frequency behavior. Table 3 gives details on the formulations of the different systems of circuit equations from a performance point of view.

The table gives details for three of the problems, T1, T3, and T5, when formulating the circuit equations using the Nodal Analysis (NA) or Modified Nodal Analysis (MNA) formulation in the time or the frequency domain. The table also shows several interesting results. For example, the formulations of the circuit equations are more time consuming in the frequency domain. This is expected since the equations involve complex numbers. Further, we see that the MNA formulation is always faster than the NA formulation even if the equation systems are larger in size. This is mainly due to an efficient formulation of (5) for which the inversion of the coefficient of potential matrix P is avoided [10].

Table 1: Reactor characteristics.

Test	Number of surface cells (N_ϕ) (charge basis function)	Number of volume cells (N_i) (current basis function)	Number of unknowns $(N_\phi + N_i)$ (total)	Number of nodes
T1-400	1 300	1 040	2 340	1 105
T2-900	2 600	2 340	4 940	2 405
T3-601	3 640	4 940	8 580	3 250
T4-605	10 920	18 460	29 380	9 750
T5-609	18 200	31 980	50 180	16 250

Table 2: Partial element calculations times for NA-implementation.

Number of processors	Time for TD-tests [s]				
	T1	T2	T3	T4	T5
Serial	9.0	35.0	81.0	-*	-
1	6.0	23.4	53.5	590.2	-
2	3.1	12.0	27.0	291.6	-
4	1.6	6.1	13.6	144.2	476.5
8	0.8	3.1	7.1	78.2	229.6
20	0.4	1.3	3.0	33.2	91.3
30	0.3	0.8	2.1	22.3	71.2

C. Total PEEC-Model Solution Time

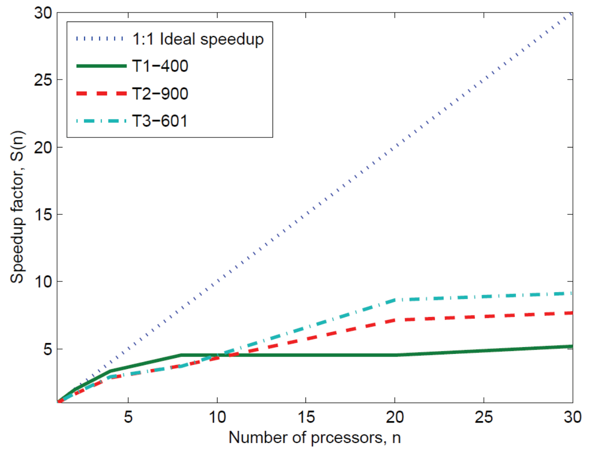
To conclude the two previous subsections, speed-up factors for total PEEC-model solutions are given. This is shown in Fig. 5 for the time domain implementation of the NA and MNA methods and in Fig. 6 for the frequency domain implementation of the NA and MNA methods.

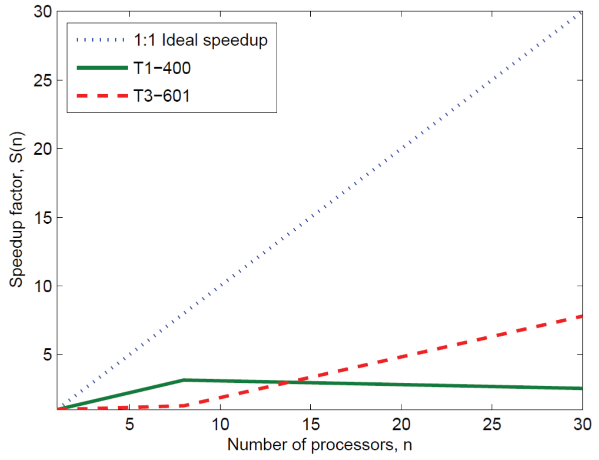
Since the speed-up factors, as presented in Figs. 5 and 6, are based on results for one processor, as seen in (7), it is not possible to show results for all test cases. However, using 30 processors, the only tests that could not be carried out with the current implementation are T4 and T5 in the frequency domain using the MNA method.

From the figures, several conclusions can be drawn:

- For small problems in the time domain, i.e. T1, and T2, increasing the number of processors does not improve the overall solution time since the communication

(a)



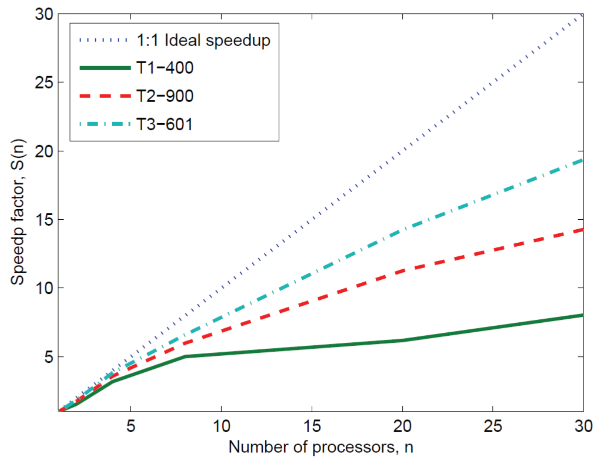


(b)

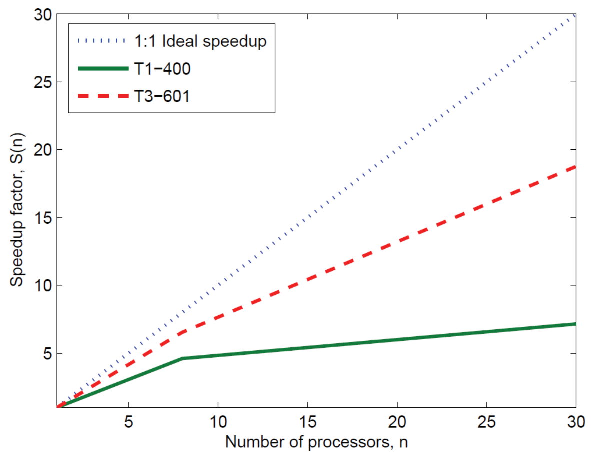
Fig. 6. Total PEEC-model solution time for frequency domain simulations using Nodal Analysis (a) and Modified Nodal Analysis (b).

Table 3: Time for formulation of circuit equations.

	Time in TD / FD [s]					
	T1-400		T3-601		T5-609	
	NA	MNA	NA	MNA	NA	MNA
Coefficient matrix size	1 300 × 1 300	2 340 × 2 340	2 640 × 2 640	8 580 × 8 580	18 200 × 18 200	50 180 × 50 180
Number of processors						
1	1.0 / 3.1	0.93 / 2.7	51.1 / 182.7	26.8 / 35	- / -	- / -
8	0.25 / 0.58	0.22 / 0.35	11.75 / 27.7	4.5 / 9.0	1 434 / 5 400	473 / -
30	0.26 / 0.32	0.14 / 0.26	4.11 / 9.7	2.0 / 3.0	634 / 2 300	138 / 380



(a)



(b)

Fig. 6. Total PEEC-model solution time for frequency domain simulations using Nodal Analysis (a) and Modified Nodal Analysis (b).

time between the processors increases and exceed the total solution time. For example in Fig. 5 (a) and (b), problem T1 saturates at 8 processors. Hence, using a bunch of processors for a small problem will not necessarily improve the performance.

- Frequency domain problems experience a larger speed-up factor compared to time domain problem. However, in general frequency domain problems are more time consuming in absolute numbers.
- In both domains, the MNA-based solver shows better speed-up factor compared to the NA counterpart. This means that in both NA figures, the problem is already saturated or the speed-up factor grows very slowly when more processors are allocated. Thus, the MNA-formulation is more suited for parallelization using ScaLAPACK.

The figures presented in this section do not reveal the absolute solution time of the problems. It might seem that the MNA formulation is the fastest. However, in fact, the NA formulation is the fastest solution method for all tests. The MNA formulation is preferred for reasons given in Sec. IV-B.

D. Memory Usage

In this section, the memory usage of the parallel solver is presented based on the following aspects:

- memory usage as a function of number of processors;

- memory usage as a function of problem size.

1) *Memory usage as a function of the number of processors*: The total memory usage M should be a function of the number of processors, n, and the problem size. This can be expressed as

$$M = an + b\,, \tag{8}$$

where *a* and *b* are constants. Figure 7 shows the total memory usage *M* as a function of processors. The increase is clearly linear and eq. (8) is verified.

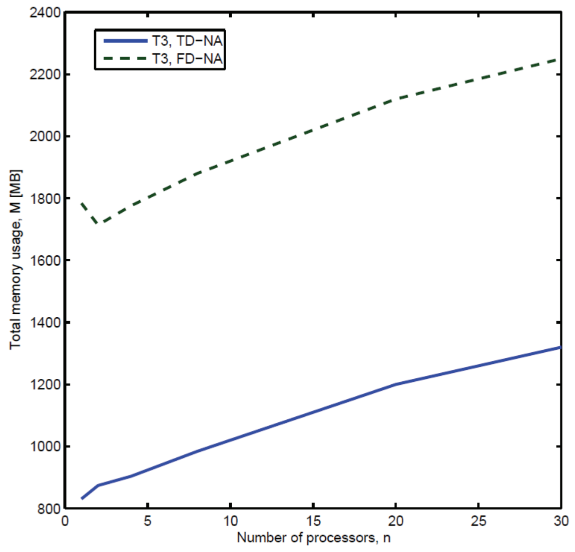


Fig. 7. Total memory usage as a function of processors.

Another way of expressing this is as average memory usage per processor *m* as

$$m = {}^{b}\!/_{n} + c\,. \tag{9}$$

This behavior for the parallel PEEC solver is verified by Table 4.

2) *Memory usage as a function of problem size*: The expected memory usage is

$$M = bn_u^2 + cn_u + d\,. \tag{10}$$

This is a simplification as in reality the actual increase is not as straightforward. This is due to the number of surface cells, volume cells, and nodes vary and the memory usage will be a more complicated function of these variables. The validity of (10) is exemplified in Fig. 8 for the parallel implementation using the NA approach in both the time and frequency domain on all the test cases from Table 1. Studying Fig. 8 when letting M be total memory usage and $(n_u = N_\phi + N_i)$ be the number of unknowns in eq. (10), the assumption is valid.

Table 4: Average memory usage for varying number of processors. Simulation of reactor.

Number of processors	Average memory usage/proc. [MB]			
	T2-900		T5-609	
	TD	FD	TD	FD
1	281	643	-	-
2	160	303	-	-
4	86	159	7 472	-
8	51	88	3 815	-
20	28	44	1 543	3 545
30	24	34	1 049	2 395

V. NUMERICAL TEST (II) – SURGE TEST

In order to test the parallel solver and to find out how large problems can be solved, a second numerical example is presented. The setup is a surge pulse that is applied to a 100_100_150 cm enclosure. This is a problem that requires a fine mesh for a large structure and a long simulation time for the pulse to decay. In this test, the enclosure is excited on the top surface with a surge pulse given by

$$i(t) = I_0(e^{-\alpha t} - e^{-\beta t})\,, \tag{11}$$

Where
$I_0 = 218810, \alpha = 11354$ and $\beta = 647265$.

The enclosure is grounded using 1Ω resistor at the bottom surface. For comparison, CST software has been used to study the same problem. Figure 9 (a) shows voltage distribution in the enclosure due to the surge test and Fig. 9 (b) a comparison of CST and PEEC results for the resistor current/voltage (due to 1Ω resistor). As can be seen, the results are close to overlapping.

The simulation time was the main advantage of the PEEC simulations for this case. It was about 31 hours with the CST (note that the CST has more number of points than PEEC simulation).

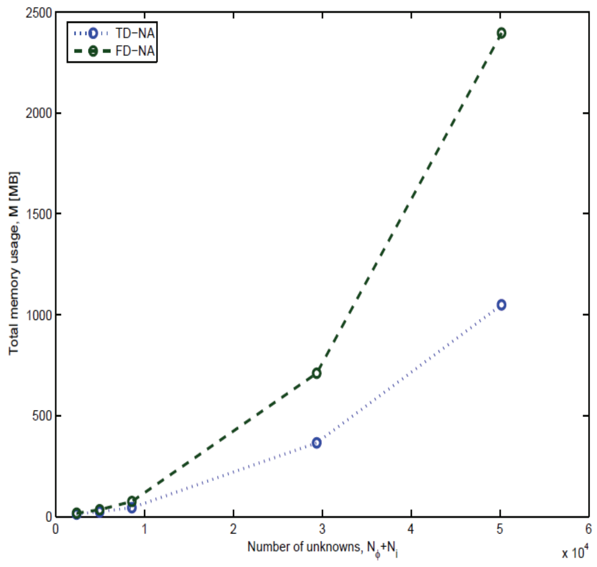
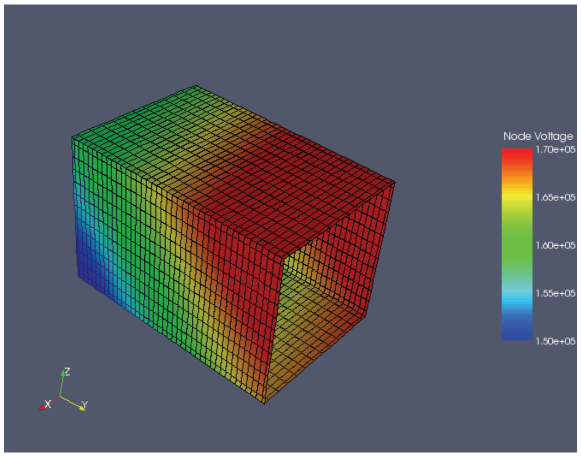


Fig. 8. Total memory usage for the Nodal Analysis implementation when increasing the problem size for a fixed number of processors (30).

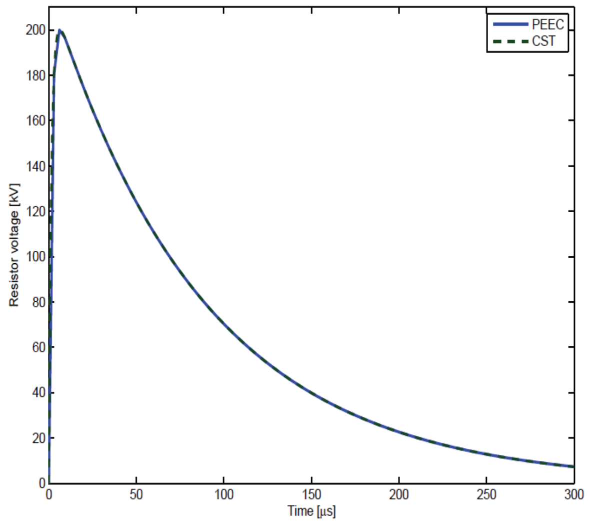
The corresponding time with PEEC was about ten minutes using a sequential code (not the parallel implementation) for a mesh corresponding to ten cells per wavelength. The comparison can seem unfair, as often when comparing different basic formulations in computational electromagnetics. This example is an optimal PEEC-type of problem due to the thin, metallic walls of the enclosure located in free space. However, CST was used to validate the results and not to benchmark the implementation.

To test the parallel implementation, three cases with different discretization level were tested using the NA, time domain solver. Table 5 describes each test case and the corresponding number of unknowns. As can be seen, Surge3 contains more than a quarter of a million of unknowns and is the largest PEEC problem that has been solved using a general PEEC implementation capable of handling time and frequency domain analysis from DC to a maximum frequency given by the mesh. These problems have been designed to be large enough to stress the solver in both memory and computational complexity sense.

During these tests, it was always considered to choose an optimum number of processors, while using many processors for a small problem is not efficient and can even degrade the performance if the solver reaches or even pass saturation point. By optimum, it is meant smallest number of processors which can provide enough memory for a problem and solve the problem as fast as possible. Table 6 shows the simulation time, number of unknowns (repeated for convenience), memory consumption, and number of allocated processors for each test case.



(a)



(b)

Fig. 9. Surge voltage simulation result at 1.8μs after impulse test (a) and comparison of PEEC and CST resistor current (b).

Table 5: Surge test characteristics.

Test	Number of surface cells (charge basis function)	Number of volume cells (current basis function)	Number of unknowns (total)	Number of nodes
Surge 1	10 404	20 400	30 804	10 200
Surge 2	48 884	96 880	145 764	48 400
Surge 3	89 244	177 240	266 484	88 800

Table 6: Total solution times and memory for NA-implementation.

Test	Time [s]	Memory [GB]	Unknowns	Processors
Surge 1	45	16	30 804	80
Surge 2	677	278	145 764	120
Surge 3	8 700	931	266 484	440

With the number of unknowns handled in the largest test case, > 250 000, many complex problems can be studied using the PEEC method. For example, pure inductance and capacitance calculations for geometrically complex geometries, R-L-C equivalent circuit extraction, shielding and radiation problems of new computational complexity. However, with more complex models, new challenges arise in order to ensure correct simulation results [29], optimal mesh generation, and optimal usage of computational resources.

VI. CONCLUSION

In this paper the first parallel PEEC-based solver was presented which is applicable to problems formulated in both the time and frequency domain solving problems from DC to the highest frequency given by the appropriate mesh. The parallel PEEC-based solver is a ported version of the original sequential PEEC-based solver which uses GMM++ linear algebra package and is suitable for systems with shared memory structure such as multi-core machines. The parallel solver is designed to run on clusters with distributed memory architecture by using ScaLAPACK package to take advantage of any number of processors allocated in a parallel computer system.

The presented implementation opens up new doors for the solution of large problems formulates using the PEEC approach. In order to understand how parallel PEEC-based solver can improve solving problems, several tests were done. The results of these tests conclude that:

- Using a number of processors for small problems will not improve the solution time and even can degrade the performance, due to saturation.
- In general, frequency domain problems need more memory and are more time consuming, compared to time domain problems, but experienced larger speed-up factors when the number of processors was increased.
- MNA-based solver in both time and frequency domain resulted better in speed-up factor than NA-based.

By using 440 processors, problems with over quarter of a million unknowns could be studied using the nodal analysis formulation in the time domain. The next step for the presented parallel implementation is to apply acceleration algorithms, e.g. FMM, MLFMM, QR and using other LAPACK-based computational libraries, making the program suitable for solving different classes of EM problems on desktop machines.

REFERENCES

[1] W. C. Chew, J. M. Jin, C. C. Lu, E. Michielssen, and J. M. Song “Fast solution methods in electromagnetics”, *IEEE Transactions on Antennas and Propagation*, vol. 45, no. 3, pp. 533–543, 1997.

[2] N. Engheta, W. D. Murphy, V. Rokhlin, and M. S. Vassilou, “The fast multipole method (FMM)”, *PIERS*, July 1991.

[3] S. Kapur and D. Long., "IES: A fast integral equation equation solver for efficient 3-dimensional extraction," *Int. Conf. on Computer Aided Design*, pp. 448–455, November 1997.

[4] Y. Liu and J. Yuan, "A finite element domain decomposition combined with algebraic multigrid method for largescale electromagnetic field computation", *IEEE Transactions on Magnetics*, vol. 42, no. 4, pp. 655– 658, April 2006.

[5] T. Hanawa, M. Kurosawa, and S. Ikuno, "Investigation on 3-D implicit FDTD method for parallel processing", *IEEE Transactions on Magnetics*, vol. 41, no. 5, pp. 1696– 1699, May 2005.

[6] A. Rubinstein, F. Rachidi, M. Rubinstein, and B. Reusser, "A parallel implementation of NEC for the analysis of large structures", *IEEE Transactions on Electromagnetic Compatibility*, vol. 45, no. 2, pp. 177–188, May 2003.

[7] A. E. Ruehli, "Equivalent circuit models for three dimensional multiconductor systems", *IEEE Transactions on Microwave Theory and Techniques*, vol. MTT-22, no. 3, pp. 216–221, March 1974.

[8] A. E. Ruehli, "Inductance calculations in a complex integrated circuit environment". *IBM Journal of Research and Development*, vol. 16, no. 5, pp. 470–481, September 1972.

[9] A. E. Ruehli and P. A. Brennan, "Efficient capacitance calculations for three-dimensional multiconductor systems", *IEEE Transactions on Microwave Theory and Techniques*, vol. MTT-21, no. 2, pp. 76–82, February 1973.

[10] G. Antonini, J. Ekman, and A. Orlandi, "Full wave time domain PEEC formulation using a modified nodal analysis approach", *Proc. of EMC Europe*, 2004.

[11] F. Monsefi and J. Ekman, "Optimization of PEEC based electromagnetic modeling code using grid computing", *Proc. of EMC Europe*, 2006.

[12] J. Ekman and P. Anttu, "Parallel implementation of the PEEC method", *Proc. of Special Session at the IEEE Int. Symp. On EMC*, 2007.

[13] J. Choi, J. J. Dongarra, R. Pozo, and D. Walker. "ScaLAPACK: A scalable linear algebra library for distributed memory concurrent computers", *Proceedings of the Fourth Symposium on the Frontiers of Massively Parallel Computation, IEEE Computer Society Press*, 1992.

[14] G. Antonini, "Fast Multipole Formulation for PEEC Frequency Domain Modeling", *Journal of Applied Computational Electromag Society,* vol. 17, no. 3, November 2002.

[15] G. Antonini, J. Ekman, A. Ciccomancini Scogna, and A. E. Ruehli, "A comparative study of PEEC circuit elements computation", *Proc. of the IEEE International Symposium on EMC*, 2003.

[16] G. Antonini, A. Orlandi, and A. Ruehli, "Speed-up of PEEC method by using wavelet transform", *Proc. of the IEEE Int. Electromagnetic Compatibility*, August 2000.

[17] G. Antonini, A. Orlandi, and A. Ruehli, "Fast Iterative Solution for the Wavelet-PEEC Method", *Proc. of the International Zurich Symposium on Electromagnetic Compatibility*, February 2001.

[18] G. Antonini and A. Orlandi, "Computational properties of wavelet based PEEC analysis in time domain", *Proc. of Applied Computational Electromagnetics Society Conference*, March 2000.

[19] A. Ruehli, D. Gope, and V. Jandhyala, "Block partitioned gaussseidel PEEC solver accelerated by QR-based coupling matrix compression techniques", *Digest of Electr. Perf. Electronic Packaging*, vol. 13, pp. 325–328, October, 2004.

[20] S. Ramo, J. R. Whinnery, and T. Van Duzer, *Fields and Waves in Communication Electronics,* John Wiley and Sons, 1994.

[21] A. E. Ruehli, G. Antonini, J. Esch, A. Mayo J. Ekman, and A. Orlandi, "Non-orthogonal PEEC formulation for time and frequency domain EM and circuit modeling", *IEEE Transactions on Electromagnetic Compatibility*, vol. 45, no. 2, pp. 167–176, May 2003.

[22] C. Ho, A. Ruehli, and P. Brennan, "The modified nodal approach to network analysis", *IEEE Transactions on Circuits and Systems*, pp. 504–509, June 1975.

[23] LTU/UAq PEEC solver. Available. Online: http://www.csee.ltu.se/peec.

[24] A. Musing, J. Ekman, and J. W. Kollar, "Efficient calculation of non-orthogonal partial elements for the PEEC method". *IEEE Transactions on Magnetics*, 45(3), March 2009.

[25] J. Ekman, G. Antonini, G. Miscione, and P. Anttu, "Electromagnetic modeling of automotive platforms based on the PEEC method". *Proc. of Applied Computational Electromagnetics Society Conference*, Verona, IT, March 2007.

[26] D. Daroui. "Performance of integral equation based electromagnetic analysis software on parallel computer systems", Master's thesis, University of Gothenburg, February 2007.

[27] J. J. Dongarra and D. W. Walker. "The design of linear algebra libraries for high performance computers", Technical Report ORNL/TM-12404, University of Tennessee, Knoxville, TN, USA, 1993.

[28] M. Enohnyaket and J. Ekman. "Analysis of air-core reactors from dc to very high frequencies using PEEC models", *IEEE Transactions on Power Delivery*, vol. 24, no. 2, April 2009.

[29] J. Ekman, G. Antonini, A. Orlandi, and A. E. Ruehli, "The impact of partial element accuracy on PEEC model stability", *IEEE Transactions on Electromagnetic Compatibility*, vol. 48, no. 1, pp. 19–32, March 2006.