CPU Based Parallel Computation of Electromagnetic Transients For Large Scale Power Systems

A. Abusalah, O. Saad, J. Mahseredjian, U. Karaagac, L. Gerin-Lajoie, I. Kocar

***Abstract*— This paper presents the implementation of a parallel sparse matrix solver for improving the computational speed of an electromagnetic transients (EMTs) simulation software. The new method is established on the KLU sparse matrix solver which is suitable for circuit based simulation methods. The solver is programmed using parallelization through automatic detection of sparse matrix submatrices separated by the natural decoupling available in transmission line/cable models. The proposed approach is demonstrated in an EMT-type software that uses a fully iterative solution method for all nonlinear models. Furthermore, it is demonstrated for a realistic large scale grid, the actual Hydro-Quebec power transmission network.**

***Index Terms*—Electromagnetic transient, modified-augmented-nodal-analysis, KLU, sparse matrix solver, parallel programming.**

I. INTRODUCTION

Computation time is a crucial parameter in the simulation of power system electromagnetic transients (EMTs). This aspect is becoming increasingly important with modern power systems that include the integration of wind generators, HVDC transmission links and various other devices. Moreover, due to the much superior accuracy of the circuit-based approach in EMT computation methods, there is a trend to extend its application to the simulation of electromechanical transients for the same data set. This could require modeling very large scale networks. EMT computations with such a network are presented and compared in [1].

It is possible to improve the computational performance for off-line EMT-type solvers by programming more efficient solution methods and models, but such research does not allow to achieve significant gains due to the inherent algorithms for circuit based modeling. Other approaches for improving performance include multiple time-step (multi-rate) solutions [2], waveform relaxation [3][4], combinations of different time-frame methods [5] and interfacing with frequency dependent network equivalents [5][6]. The main difficulty with such methods is generalization, automation and control of accuracy. The industrial grade implementation of such methods into existing EMT-type software poses major challenges.

A direct approach for off-line EMT-type computational speed improvement is the application of parallelization. This is supported by the fact that the current trend in the computing industry is to deliver parallel computers rather than faster processor units.

The parallelization approach is researched in many publications [7][8] and has been initially applied in real-time simulation tools [9]-[11]. Off-line methods have been proposed in [12][13] and other publications. Network tearing for parallelization without any loss of accuracy is based on the natural time delay formed by distributed parameter transmission line (or cable) models. It is also possible to avoid approximations using other tearing techniques, such as in [14], when transmission line models are not present in a given network.

In addition to CPU based parallelization, work has been done using other technologies, such as GPU [15].

An important difficulty in several references presented above, such as [13], is that user intervention is required to decide on parallelization tasks, interfacing procedures or selection of equivalents. Some real-time simulators [9] are capable of automatic task scheduling, but they are not based on the sparse-matrix solution approach researched in this paper.

The objective in this paper is to present shared memory CPU based parallelization on conventional multi-core computers. The objective is also to avoid any user intervention in the parallelization process. The presented work targets the upgrading of existing sparse matrix based solvers. Network tearing for parallelization is based on the natural decoupling delay caused by distributed parameter transmission line models. The are no approximations in the proposed approach.

This paper demonstrates the application of a new sparse matrix solver for an existing EMT-type simulation tool (EMTP [16]) for improving computational performance through automatic parallelization. Another distinctive contribution in this paper is that parallelization is applied for a fully iterative solver. All nonlinear models are solved simultaneously using the Newton method. The iterations are essential for delivering highest accuracy, but the iterative process creates supplementary computational burden. The

A. Abusalah, J. Mahseredjian and I. Kocar are with Polytechnique Montréal, Campus Université de Montréal, 2900, Édouard-Montpetit, Montréal (Québec), Canada, H3T 1J4 (e-mail: jeanm@polymtl.ca).
O. Saad is with IREQ, Hydro-Québec, Canada
U. Karaagac is the Hong Kong Polytechnic University
L. Gérin-Lajoie is with Hydro-Québec, Canada
Paper submitted to the International Conference on Power Systems Transients (IPST2017) in Seoul, Republic of Korea June 26-29, 2017

contributions are tested on a new version of the very large scale Hydro-Quebec grid benchmark initially presented in [1]. The benchmark is used directly without any intervention into its topology and model selections for assisting the parallelization process.

The first section of this paper recalls the solution methods used in EMTP. The second section presents the selection and implementation of a new sparse matrix solver. The last section presents computing timings for the Hydro-Quebec benchmark.

II. EMTP SOLUTION METHODS

In EMTP the model equations are assembled using modified-augmented nodal analysis (MANA). Bold characters are used below to denote matrices and vectors.

At each time-point the solved system of equations is given by [16][17]

$$\mathbf{Ax} = \mathbf{b} \tag{1}$$

In its expanded form, this system of equations is written as

$$\begin{bmatrix} \mathbf{Y_n} & \mathbf{A_c} \\ \mathbf{A_r} & \mathbf{A_d} \end{bmatrix} \begin{bmatrix} \mathbf{v_n} \\ \mathbf{i_x} \end{bmatrix} = \begin{bmatrix} \mathbf{i_n} \\ \mathbf{v_x} \end{bmatrix} \tag{2}$$

where the classic nodal admittance matrix $\mathbf{Y_n}$ is augmented with model equations written in the row matrix $\mathbf{A_r}$ and the coefficient matrix $\mathbf{A_d}$ for the supplementary unknowns. The matrix $\mathbf{A_c}$ is used for linking the model currents with models expressed using nodal analysis. The unknowns are the nodal voltages $\mathbf{v_n}$ and model currents $\mathbf{i_x}$. The right hand side variables are the nodal current injections $\mathbf{i_n}$ and model voltages $\mathbf{v_x}$ in the augmented part. As explained in [17], MANA formulation is more flexible than classic nodal analysis and simplifies the inclusion of model equations. Ideal and non-ideal switch equations, for example, are included directly in the augmented part. Other variables can be used in addition to currents in $\mathbf{i_x}$.

The system of equations (1) is solved at each time-point. Nonlinear model equations are included using linearization at each time-point. At any time-point the linearized equation of a nonlinear model *k* can be expressed as [17]

$$i_k = y^{(j)} v_k + I_Q^{(j)} \tag{3}$$

where the slope $y^{(j)}$ and the intercept current $I_Q^{(j)}$ are found from the model equation differentiation at the operating voltage point at each iteration *j*. The above equation is generic and can be also written in its matrix form more complex device models.

In EMTP, the matrix $\mathbf{A}$ is saved and solved in its non-symmetric form. This is important for accommodating non-symmetric model equations. EMTP uses a sparse direct solver that is based on LU decomposition with minimum degree ordering [18]. The sparse LU solver has an ordering step to minimize fill-in, symbolic factorization step to determine the non-zero pattern and the computation step of $\mathbf{L}$ and $\mathbf{U}$ matrix factors. During the numeric elimination stage the pivotal permutation of $\mathbf{A}$ is found and the matrix is permuted in order to avoid numerical instabilities. The minimum degree ordering technique is used in the current version of EMTP to minimize the number of factors. Once the $\mathbf{LU}$ factors are found, backward and forward substitution is performed to find the solution vector $\mathbf{x}$ based on the right hand side $\mathbf{b}$.

Any slope change in (3) or any change in switch status requires updating the matrix $\mathbf{A}$ in the iterative process at each solution time-point until convergence within prescribed tolerance. Although a very efficient iterative process is programmed, it is still required to recalculate the matrices $\mathbf{L}$ and $\mathbf{U}$. This refactorization process increases the computational burden.

III. REPLACEMENT OF SPARSE MATRIX SOLVER

The KLU [19][20] sparse matrix solver is a direct solver optimized for the solution of sparse electrical circuit equations. It has been demonstrated [21] to provide better performance than other sparse matrix packages for highly sparse matrices. KLU can be used to solve the non-symmetric matrices of EMTP.

KLU is a sparse solver that is developed based on matrix graph theory and uses a combination of sorting techniques and factorization techniques to solve any sparse matrix in the least computing time, and provide accurate and reliable results. Like many other solvers, this solver consists of three main stages. The three stages are symbolic analysis, numerical factorization and solve.

During the symbolic analysis stage, the matrix undergoes multiple stages of permutation that guarantee re-writing the matrix in its block diagonal form (BTF) when distributed parameter transmission line models exist in the simulated network. This form allows to divide the matrix into multiple submatrices that are independent of each other and can be solved independently. In order to achieve the BTF format, two permutation matrices are computed and applied on the matrix $\mathbf{A}$, namely the row permutation $\mathbf{P_R}$ and the column permutation $\mathbf{P_C}$ matrices. The resulting BTF matrix is given by

$$\mathbf{A_{BTF}} = \mathbf{P_R A P_C} \tag{4}$$

Row permutation is found by applying a first depth search in order to allocate all strongly connected nonzero elements of the matrix that form independent blocks. Each group of strongly connected elements represents an independent submatrix of $\mathbf{A}$. The column permutation matrix is computed by finding the maximum transverse of the directed graph that represents the matrix $\mathbf{A}$.

Fig. 1 shows a simple network with two subnetworks interconnected via a distributed parameter transmission line. The nonzero pattern of the matrix $\mathbf{A}$ in (1) of the this system is presented in Fig. 2. It can be observed that $\mathbf{A}$ does not initially have the BTF.

Applying the permutation matrices $\mathbf{P_R}$ and $\mathbf{P_C}$ transforms the matrix of Fig. 2 into its BTF of Fig. 3. Now the permuted

matrix has two independent blocks along its diagonal, and these two blocks represent the two subnetworks connected through the constant parameter transmission line model. As it is well known, this procedure is completely automatic and does not require user intervention. There is no need to program topological analysis for discovering the subnetworks separated by transmission lines. The number of subnetworks (independent blocks) is fixed by the number of transmission line models connected through any topology.

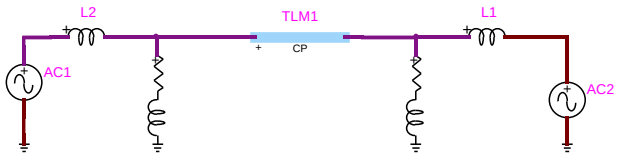

Fig. 1 Simple test case for demonstrating BTF permutation

The sparsity pattern of $\mathbf{P_R}$ for this case is presented in Fig. 4. It works as an adjacency matrix (only ones and zeros). The matrix $\mathbf{P_C}$ is the transposed version of $\mathbf{P_R}$.

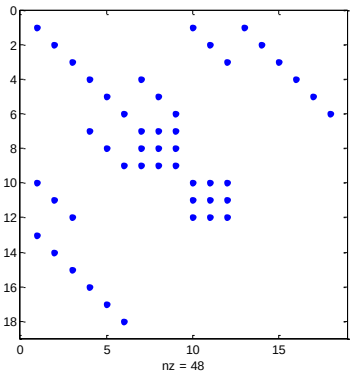

Fig. 2 Nonzero pattern of network shown in Fig. 1

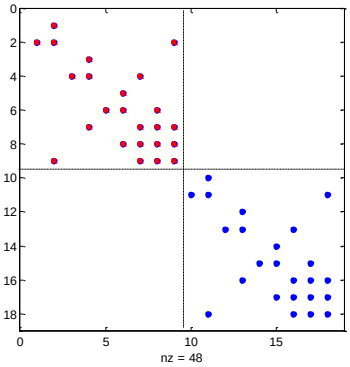

Fig. 3 Nonzero pattern of network shown in Fig. 1, after applied BTF

KLU uses fill-in reduction techniques to reduce the number of nonzeros in **L** and **U** matrices. In this paper, Approximate Minimum Degree Ordering (AMD) has been used as the default ordering technique in KLU. It is worth mentioning that the symbolic analysis step and the BTF permutation are performed only once at the beginning of the simulation process. This is due to the fact that the network structure and hence the strongly connected components of **A** do not change during the entire simulation period. As for the refactorization process, the pivoting strategy used in KLU is more efficient than in the current EMTP package and consequently reduces the computing time associated to nonlinear models and switching devices.

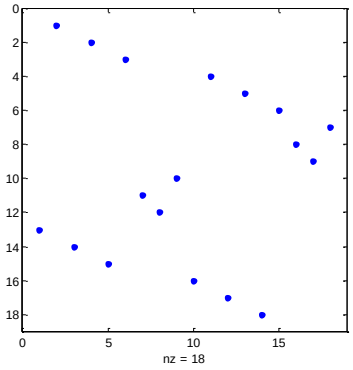

Fig. 4 Matrix $\mathbf{P_R}$ used for Fig. 3

During the second stage of KLU, the nonzero pattern of $\mathbf{LU}$ is calculated using Gilbert-Peierls' algorithm [19]. Once the nonzero pattern is found, a left looking numerical factorization with partial pivoting is performed to calculate the numerical values of **L** and **U** to transform $\mathbf{A_{BTF}}$ into

$$\mathbf{A'_{BTF}} = \mathbf{LU} \tag{5}$$

The third step of KLU is the solution step during which forward and backward substitutions are conducted in order to obtain the results for vector $\mathbf{x}$.

IV. PARALLEL KLU IMPLEMENTATION

In order to accelerate the performance of the KLU solver, it is necessary to implement it using multiple cores in parallel. The OpenMP [22] multithreading approach can be used for that purpose. OpenMP is an API for multi-platform shared-memory parallel programming [23]. It reduces the overall parallel programming effort. OpenMP requires the user to define different segments of the code where parallel processing is required, using directives. Once all parallel segments are specified, all parallization processes such as thread launching, control, synchronization and termination are done by the compiler.

The OpenMP directives allow to access physical cores and perform hyperthreading.

In this paper, parallel programming has been applied on the factorization and substitution steps only, whereas the symbolic stage is kept purely sequential due to the fact that it is done only once at the beginning of simulations. Depending on the network size, the number of threads and the number of CPU cores in a given computer, the $\mathbf{A_{BTF}}$ blocks are automatically distributed to different threads in order to be factorized and solved.

V. TEST CASES

The test cases presented here are based on the Hydro-Quebec grid. There are two versions, as described in [1]: L-Network and R-Network. The network sizes and contents have increased since the presentation of [1]. The L-Network refers

to the very large version and the R-Network is a reduced version created from the L-Network. In this paper both network versions are solved directly in parallel without any user-intervention and without any topological analysis for helping parallelization. Network partitioning is based solely on the BTF algorithm using distributed parameter line models. Only constant parameter (CP-lines) line models are used. PI-section models are used for shorter lines to avoid penalizing the numerical integration time-step upper bound.

The simulation results are exactly the same, with and without parallelization since there are no approximations. All simulations are performed on a Intel (i7-4900MQ) computer with 4 cores (8 threads).

A. Hydro-Quebec L-Network version

As explained in [1] the very large scale L-Network constitutes a reference for obtaining network equivalents for various study purposes. It is also a unified environment for maintaining and extracting data for various applications.

The latest network contents (main devices) are as follows:

- Size: 94706 devices, 42474 power devices and 52232 control diagram blocks
- Circuit nodes: 29797
- 355 CP-lines, 904 PI-sections
- 2098 3-phase transformers
- 174 zinc oxide arresters (nonlinear)
- 213 nonlinear inductances
- 3663 ideal switches
- 916 load models for a total of 40.5 GW
- 10 SVC models (average-value version)
- 349 synchronous machine models with AVR and governor models for a total of 43 GW of generation
- Matrix **A** size: 41797×41797

The sparsity pattern of the matrix **A** is presented in Fig. 5. The corresponding BTF version is presented in Fig. 6. There are a total of 181 blocks, the largest block size is 13584×13584 and smallest block is 9×9.

Due to the presence of nonlinearities, the average number of iterations per time-point is 2. The numerical integration time-step is $\Delta t = 50\mu s$. The performed simulation is a fault case and the simulation interval is 10 s.

The computer timings are presented in Table I. EMTP-1 is referring to the sparse matrix package that has been replaced by the KLU package in EMTP-KLU. It is apparent that the EMTP-KLU approach is 1.15 times faster than EMTP-1 on a single thread. On 2 threads the gain reaches 1.5 times. There is no increase in gains after 2 threads. This is due to overhead by thread and data management. The main difficulty in this case is that the large block of 13584 rows becomes the limiting factor. Since it represents a dense network region, it does not include distributed parameter transmission lines that can be simulated with the given Δt lower bound. It is also obvious that reducing Δt has a negative impact on computational performance.

B. Hydro-Quebec R-Network version

The reduced version of the L-Network is the R-Network. As explained in [1], this reduction allows to extract a portion of the network for faster simulations.

The latest network contents (main devices) are as follows:

- Size: 21275 devices, 7141 power devices and 14134 control diagram blocks
- Circuit nodes: 4186
- 118 CP-lines, 249 PI-sections
- 149 3-phase transformers
- 135 zinc oxide arresters (nonlinear)
- 50 nonlinear inductances
- 144 ideal switches
- 390 load models for a total of 35 GW
- 10 SVC models (average-value version)
- 56 synhronous machine models with AVR and governor models for a total of 37 GW of generation
- Matrix **A** size: 5272×5272

The sparsity pattern of the matrix **A** is presented in Fig. 7. The corresponding BTF version is presented in Fig. 8. There are a total of 77 blocks, the largest block size is 924×924 and smallest block is 3×3.

The average number of iterations per time-point is 2. The computer timings are presented in Table II for $\Delta t = 50\mu s$. Now it is possible to achieve gains up to the 4th thread for a maximum of 1.36.

In this case it is possible to increase to $\Delta t = 100\mu s$ which reduces the timings in Table II by approximately half.

Table I Computer timings (s), L-Network

EMTP-1	**EMTP-KLU**		
	1 thread	2 threads	4 threads
7444	6426	4995	5526

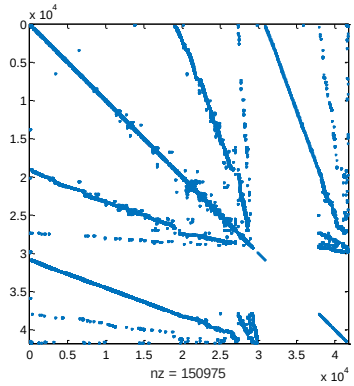

Fig. 5 Hydro-Quebec grid L-Network version, sparse matrix

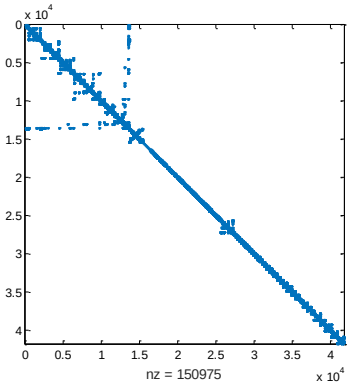

Fig. 6 Hydro-Quebec grid L-Network version, sparse matrix in BTF

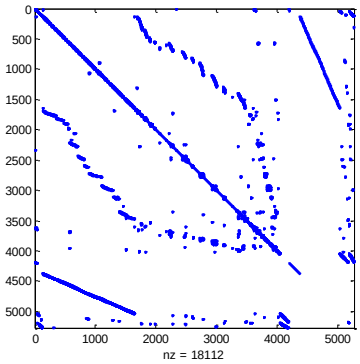

Fig. 7 Hydro-Quebec grid R-Network version, sparse matrix

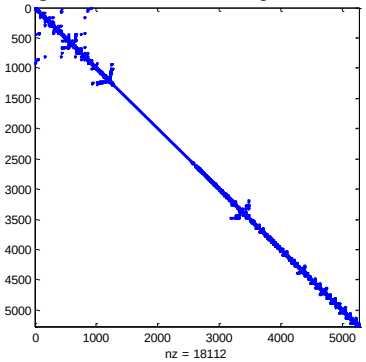

Fig. 8 Hydro-Quebec grid R-Network version, sparse matrix in BTF

Table II Computer timings (s), R-Network

EMTP-1	**EMTP-KLU**		
	1 thread	2 threads	4 threads
188	168	146	138

VI. Conclusion

This paper presented the implementation of a parallel sparse matrix solver for improving the computational speed of an electromagnetic transients simulation software. The presented method is based on the KLU sparse matrix solver. The solution process is programmed using network partitioning through the natural decoupling formed by distributed parameter transmission line/cable models. The applied partitioning is automatic and does not require any user intervention. It is calculated automatically from the original network matrix by applying block diagonal factorization.

Parallelization is programmed using the OpenMP multithreading approach. A fully iterative solution is applied for the accurate solution of nonlinear models.

The purpose of this presentation was to improve the computational performance of a real large scale grid case, namely the Hydro-Quebec network model, in an EMT-type software. The target was to simulate two versions of the network without any alterations in the assembled topology and applied models.

The presented computer timings demonstrate important gains, but those gains become limited by the largest matrix blocks and thread programming overhead. It is the first time that parallelization is demonstrated for such a large network case using an EMT-type software. The practical aspects of the case demonstrate that further research is required for decreasing computing time for real cases and given the paradigm that networks should be solved directly as assembled and without any user intervention.

VII. References

[1] L. Gérin-Lajoie, J. Mahseredjian, “Simulation of an extra large network in EMTP: from electromagnetic to electromechanical transients”, International conference on power system transients, 2009, Kyoto, Japan.

[2] A. Benigni, A. Monti, and R. Dougal, “Latency-based approach to the simulation of large power electronics systems,” IEEE Transactions on Power Electronics, vol. 29, no. 6, pp. 3201–3213, June 2014.

[3] J. M. Bahi, K. Rhofir, J.-C. Miellou, "Parallel solution of linear DAEs by multisplitting waveform relaxation methods," Elsevier, Linear Algebra and its Applications, Aug. 2001, pp. 181-196.

[4] Pan European Grid Advanced Simulation and State Estimation. “Algorithmic requirements for simulation of large network extreme scenarios,” Report D4.1, April 2011.

[5] Y. Zhang, A. M. Gole, W. Wu, B. Zhang, H. Sun, “Development and Analysis of Applicability of a Hybrid Transient Simulation Platform Combining TSA and EMT Elements”, IEEE Transactions on Power Systems, vol. 28, no. 1, pp. 357-366, 2013.

[6] A. Ramirez, A. Mehrizi-Sani; D. Hussein, M. Matar, M. Abdel-Rahman, J. J. Chavez, A. Davoudi, S. Kamalasadan, “Application of Balanced Realizations for Model-Order Reduction of Dynamic Power System Equivalents”, IEEE Transactions on Power Delivery, vol. 31, no. 5, pp. 2304-2312, 2016.

[7] J. Mahseredjian, V. Dinavahi, J. A. Martinez, “Simulation Tools for Electromagnetic Transients in Power Systems: Overview and Challenges”, IEEE Transactions on Power Delivery, vol. 24, no. 3, pp. 1657-1669, 2009.

[8] D. M. Falcao, E. Kaszkurewicz, and H.L.S. Almedia, “Application of Parallel Processing Techniques to the Simulation of Power System Electromagnetic Transients”, IEEE Transactions on Power Systems, Vol. 8, Issue 1, pp. 90-96, Feb. 1993.

[9] D. Paré, G. Turmel, J.-C. Soumagne, V. A. Do, S. Casoria, M. Bissonnette, B. Marcoux, D. McNabb, “Validation tests of the Hypersim digital real time simulator with a large AC-DC network”, International conference on power system transients, 2003, New Orleans.

[10] S. Abourida, C. Dufour, J. Belanger, G. Murere, N. Lechevin, and B. Yu, “Real-time PC-based simulator of electric systems and drives”, Proc. 17th IEEE APEC, Applied Power Electronics Conf. and Expo., Mar. 10–14, 2002, vol. 1, pp. 433–438.

[11] R.Kuffel, J. Giesbrecht, T. Maguire, R. P.Wierckx, and P. G. McLaren, “RTDS-A fully digital power system simulator operating in real-time”, Proc. EMPD’95, 1995, vol. 2, pp. 498–503.

[12] R. Singh, A. M. Gole, C. Muller, P. Graham, R. Jayasinghe, B. Jayasekera, D. Muthumuni, “Using Local Grid and Multi-core Computing in Electromagnetic Transients Simulation”, International conference on power system transients, 2013, Vancouver, Canada.

[13] S. Montplaisir-Goncalves, J. Mahseredjian, O. Saad, X. Legrand, A. El-Akoum, “A Semaphore-based Parallelization of Networks for Electromagnetic Transients”, International conference on power system transients, 2013, Vancouver, Canada.
[14] C. Dufour, J. Mahseredjian and J. Bélanger, “A Combined State-Space Nodal Method for the Simulation of Power System Transients,” IEEE Trans. on Power Delivery, Vol. 26, Issue 2, April 2011, pp. 928-935.
[15] J. K. Debnath, W.-K. Fung, A. M. Gole, S. Filizadeh “Electromagnetic Transient Simulation of Large- Scale Electrical Power Networks using Graphical Processing Units”. 2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE).
[16] J. Mahseredjian, S. Dennetière, L. Dubé, B. Khodabakhchian and L. Gérin-Lajoie, “On a new approach for the simulation of transients in power systems”. Electric Power Systems Research, Volume 77, Issue 11, September 2007, pp. 1514-1520.
[17] J. Mahseredjian, U. Karaagac, S. Dennetière, H. Saad, “Numerical Analysis of Power System Transients and Dynamics: Simulation of electromagnetic transients with EMTP-RV”, A. Ametani, editor, IET Power and Energy series, pp. 103-134, 2015.
[18] S.C. Eisenstat, M.C. Gursky, M. H. Schultz, A. H. Sherman. “Yale Sparse matrix Package”.
[19] T.A. Davis, and E. P. Natarajan, ”Algorithm 907: KLU, a direct sparse solver for circuit simulations problems,” ACM Trans. Math. Softw., Vol. 37, pp 36:1-36:17, September 2010.
[20] E. P. Natarajan, “KLU-A High Performance Sparse linear solver for Circuit Simulation Problems”, Master Thesis, University of Florida, 2005.
[21] F. González, A. Luaces, D. Dopico, M. González, “Parallel Linear Equation Solvers and Openmp in the Context of Multibody System Dynamics”, Proceedings of the ASME 2009 International Design Engineering Technical Conferences, Aug. 30-Sept. 2, 2009, San Diego, California, USA, 11 pages.
[22] http://www.openmp.org/wp-content/uploads/openmp-4.5.pdf
[23] C. Szydlowski, “Multithreated technology and multi-core processors,” Infrastructure Processor Division Intel Corporation”, 2005, May 1.