Sparse Solver Application for Parallel Real-Time Electromagnetic Transient Simulations

B. Bruned, J. Mahseredjian, S. Dennetière, A. Abusalah, O. Saad

***Abstract*—The main purpose of this research is to speed-up real-time simulations of electromagnetic transients (EMTs) using sparse linear solver techniques. This paper presents the integration of a direct sparse linear solver (KLU) into a real-time software for EMT simulation. This solver is combined with parallelization of network solution. Fill-in reduction techniques are investigated as well as partial refactorization to speed-up computations. The pivoting technique during refactorization is asserted in terms of simulation stability as compared to existing sparse solver based on code generation without pivoting. Performance and validation are studied on practical power system cases with real-time Hardware-In-the-Loop (HIL) simulation. Substantial performance gains, up to 50%, are obtained using fill-in reduction and partial refactorization. Pivoting is necessary to maintain numerical stability.**

***Keywords*: EMT simulation, real-time, Hardware-In-The-Loop, Direct sparse linear solver, Parallelization, Compensation Method.**

I. Introduction

Energy Transition raises important challenges for grid operators to integrate renewable energy sources [1]. This involves more power electronics equipment in the grid and new interaction problems that must be simulated and studied. The circuit-based electromagnetic transient (EMT) simulation approach is currently increasingly used to study the integration of renewable energy sources [2]. It is able to deliver highly accurate computations. Moreover, real-time hardware-in-the-loop (HIL) EMT simulation can be used for accurate simulations with actual controller replicas [1]. The computation time is an important factor for HIL and parallelization of solution method is essential.

The traditional line-delay (LD) technique parallelization, is based on the propagation delay of transmission lines or cables. If this delay is greater than the numerical integration time-step, the network solution can be decoupled without any loss of accuracy. This technique has been implemented in both offline [3],[4] and real-time environments [5]. When there is no LD, other techniques have to be implemented. One of such techniques is the compensation method (CM) [6],[7]. In recent works [8][9] the CM has been programmed and tested in real-time mode with demonstrated advantages.

In addition to parallelization, sparse linear solvers [10] can be used to improve numerical performance. The nodal formulation of network equations involves the solution of sparse linear systems. Sparse LU decomposition is commonly used to solve such systems in EMT-type software. Previous works [4] have studied and integrated an efficient sparse LU decomposition solver named KLU [11][12], for parallel offline simulation. A modified version of KLU (MKLU) has been used to benefit from partial refactorization techniques.

The first contribution of this paper is to integrate and test MKLU into the real-time environment HYPERSIM [5], as a replacement of an existing legacy sparse solver (GenCode) based on code generation. It is the first time that such a solver is integrated into an industrial real-time simulation environment. The second contribution is to identify the most efficient sparse solver techniques to speed up real-time EMT simulation: fill-in reduction, partial refactorization and pivoting strategy. All are available in MKLU. Fill-in reduction and partial refactorization are combined for best efficiency. The third contribution is to combine these sparse solver techniques with parallelization techniques to speed up even more the simulation. Two kinds of parallelization technique are used in this paper, namely LD and CM. The performance of MKLU is compared with GenCode for practical power system cases.

This paper is organized as follows. The integration of MKLU solver is described in Section II. Through each step of the sparse linear solver, speed-up techniques are identified (fill-in reduction, partial refactorization). In Section III, a detailed comparison is performed between MKLU and GenCode on the impact of fill-in reduction, pivoting strategy and real-time performance for practical power system cases with HIL.

II. Direct sparse linear solver

Using classic nodal (modified-augmented-nodal analysis is not available in HYPERSIM) formulation with companion circuit models, network equations can be written in a linear form to be solved at each time-point:

$$\boldsymbol{Y_n v_n = i_n} \tag{1}$$

where $\boldsymbol{Y_n}$ is the admittance matrix of the network, $\boldsymbol{v_n}$ the vector of node voltages and $\boldsymbol{i_n}$ the known vector of current sources that include history term injections from companion circuit equivalents. For the nonlinear case, a linearized Norton equivalent [3][13] is provided and updated at each iteration. The linear system of (1) is generally sparse and $\boldsymbol{Y_n}$ is not necessarily symmetric. Traditionally, direct LU sparse decomposition [10] is used to solve (1). It is preferred over iterative methods [14] for performance, robustness and predictability. The nodal admittance matrix is factorized as

$$\boldsymbol{Y_n = L_n U_n} \tag{2}$$

As well known, a forward substitution is first performed

B. Bruned and S. Dennetière are with RTE, Jonage, France (e-mail of corresponding author: boris.bruned@rte-france.com). J. Mahseredjian and A. Abusalah are with Polytechnique Montréal. O. Saad is with Hydro-Québec.

Paper submitted to the International Conference on Power Systems Transients (IPST2023) in Thessaloniki, Greece, June 12-15, 2023.

$$\boldsymbol{L_n x_n = i_n} \quad (3)$$

and followed by backward substitution

$$\boldsymbol{U_n v_n = x_n} \quad (4)$$

As $\boldsymbol{Y_n}$ is sparse, a sparse matrix format is adopted, using the standard Yale format (CSC) [15].

Once the sparse matrix has been defined, the sparse linear system solution proceeds as follows: symbolic analysis, factorization, and solution (forward and backward and substitutions). The first two steps will be detailed below, along with the integration of the KLU solver into a real-time environment.

A. Symbolic analysis

The symbolic analysis step memorizes the sparse structure of the LU decomposition for the numerical factorizations. Indeed, the sparse structure of $\boldsymbol{Y_n}$ remains fixed throughout the computation steps. Ordering methods are used during this phase to minimize fill-in. Indeed, a permutation of the elements of the matrix is sought to reduce the number of non-zero values in the factorization. The obvious objective is to save time during the solution phase. Finding an ordering is to compute permutation matrices $\boldsymbol{P_n}$ and $\boldsymbol{Q_n}$ such as:

$$\boldsymbol{\widetilde{Y}_n = P_n Y_n Q_n} \quad (5)$$

where the fill-in of the LU decomposition of $\boldsymbol{\widetilde{Y}_n}$ is less than $\boldsymbol{Y_n}$.

There are two main families of ordering methods to reduce fill-in [10] : local and global methods. One the most used local methods is the minimum degree [16] ordering. It selects at each stage of Gaussian elimination the node which has a minimum number of neighbors (if the sparse matrix is seen as a graph). This method has been applied to electrical networks and showed its effectiveness in the symmetrical case. The approximate minimum degree ordering (AMD) method [17] is an improvement. Other ordering techniques have been used in solvers like COLAMD (Column Approximate Minimum Degree) [18].

Global methods are based on Nested Dissection [10] which applies the principle of divide-and-conquer heuristics. The graph associated with the sparse matrix is partitioned into sub-graphs. By following the recursive structure of the partition (binary tree), the two sub-graphs are factorized followed by the interface variables between the two graphs. Graph partitioning algorithms are used, like Metis [19] or Scotch [20]. Nested Dissection amounts to formulating the solved matrix into a bordered-block-diagonal form.

B. Factorization

Factorization is the main element of the resolution of a linear system and consists in numerically factoring the sparse matrix into the LU form. Different strategies can be chosen regarding scaling, pivoting and decomposition.

Scaling can be used to improve matrix conditioning. Scaling does not necessarily contribute in terms of accuracy and stability for the simulation of power systems. Indeed, as an example, switch modelling as R_{open}/R_{close} resistors can create a wide range of values in the admittance matrix, in which case scaling can make things worse.

The choice of pivot in the Gaussian elimination, is important for the stability of factorization. There are three strategies for pivoting. Without pivoting the elimination is processed using the matrix diagonal elements. This can cause stability issues due to errors caused by large/small values in the matrix. The pivot may become invalid (close to zero) and make the decomposition unstable.

With partial pivoting, during factorization, the selected pivot is the maximum absolute value of the considered column and allows to prevent numerical instabilities. With full pivoting, the pivot is selected based on the maximum absolute value considering rows and columns.

Full pivoting is very rarely used. It is more expensive in terms of computing times and does not necessarily provide better stability for power system matrices. Instead, partial pivoting is preferred. During factorization, it is also possible to check if the previously calculated pivot is valid. If the pivot is still valid, there is no need to modify it. Otherwise, a full factorization must be performed.

The type of LU factorization may differ between solvers. Two main factorization strategies are used. In the right-looking [21] factorization, the matrix is factorized from top left to bottom right. The left-looking approach [11][12] is more advantageous for sparse power system matrices. The matrix is factorized along the columns from left to right.

C. Partial refactorization

The following changes have been made in [4] to optimize the KLU solver (MKLU, modified KLU) for EMT simulations and parallelization.

The first one is the pivot validity test. If the pivot is no longer valid, the refactorization is stopped and a full factorization is performed. This avoids performing full factorization when the pivot remains valid in switching networks.

The second enhancement is partial refactorization. It is applied only on the values of the matrix which have changed. First, fill-in reduction ordering is applied to $\boldsymbol{Y_n}$. Then, at each time-step, the minimal column index, n_{chg}, where there is a value change in $\boldsymbol{Y_n}$, is computed. Benefiting from the left-looking LU decomposition, refactorization is only proceeded for columns between n_{chg} and n (size of $\boldsymbol{Y_n}$**)**. This technique is named, hereinafter, RefactChg.

In order to make n_{chg} as high as possible at each solution time-point, it is possible to order first the nodes which belong to linear elements and then the ones which come from time-varying elements (switches or nonlinear elements). Equation (6) below depicts this ordering.

$$\boldsymbol{Y_n} = \begin{bmatrix} \boldsymbol{Y_f} & \boldsymbol{Y_{fv}} \\ \boldsymbol{Y_{vf}} & \boldsymbol{Y_v} \end{bmatrix} \quad (6)$$

Where $\boldsymbol{Y_f}$ is the fixed part of $\boldsymbol{Y_n}$, $\boldsymbol{Y_v}$ contains the time-varying elements of $\boldsymbol{Y_n}$ and n_{chg} is determined only for $\boldsymbol{Y_v}$. This technique is named, hereinafter, RefactVar. It is noticed that it can conflict with the fill-in reduction ordering, since it can break the fill-in optimization. To resolve this conflict between the two ordering approaches (fill-in reduction and partial refactorization), RefactVarOpt method is introduced in this paper. It proceeds by creating (6) and then applying fill-in

reduction only on $\boldsymbol{Y_f}$. The efficiencies of RefactChg and RefactVarOpt are compared below.

The limitation of partial refactorization used in this section is that all columns from n_{chg} to n have to be refactorized although only few of them may need it. Possible improvements based on [22][23] are left for further works.

D. Integration into parallel real-time environment

Two parallelization techniques are considered in real-time mode. The first one is delay-based. Network analysis identifies the decoupling elements which are the power transmission lines (or cables) with propagation delays higher than the numerical integration time-step. Stublines with artificial delays can be inserted for artificial decoupling when actual lines do not exist. Then, an automatic task mapper assigns subnetworks with their control system equations to processor units [24]. Each subnetwork is solved independently using the selected sparse solver.

The second parallelization technique is CM, which has been recently tested in real-time [8][9] simulations. It does not create inaccuracies as with stublines and delivers a simultaneous solution for network equations. The Fig. 1 recalls the three steps of CM for two subnetworks (N_1 and N_2) which have been decoupled through wires. $\boldsymbol{Y_1}$ and $\boldsymbol{Y_2}$ are admittance matrices, $\boldsymbol{i_1}$ and $\boldsymbol{i_2}$ the known or historic nodal current injections, $\boldsymbol{i_c}$ is for compensation branch currents, $(\boldsymbol{Z_{th1}}, \boldsymbol{v_{th1}})$ and $(\boldsymbol{Z_{th2}}, \boldsymbol{v_{th2}})$ are the Thevenin equivalents along the cutting branches. The two parallel steps, Thevenin equivalent computations (step-1) and superposition (step-3), involve the solution of linear systems with subnetwork admittance matrices. MKLU or GenCode can be used in step-1. The sequential step-2, deals with dense impedance matrices for which LAPACK [21] can be used. In this paper, the GenCode solver is preferred over LAPACK for performance, with only non-zero operations printed in the generated code.

The integration of MKLU into the real-time environment is done as follows. First, MKLU is pre-compiled as a static library on the real-time simulator. Then, its include files allow to call its functions and use its data structures directly on the generated code of the simulated network. Also, a standard Yale format (CSC) is used to represent sparse matrices in the simulation code. Finally, the simulation code is compiled, linking with the static MKLU library.

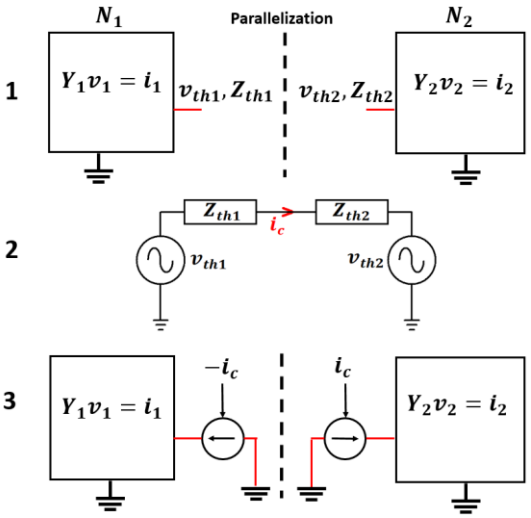

Fig. 1. Overview of CM steps for parallelization.

III. Performance and Validation Analysis

Two sparse linear solvers are tested in this section: the legacy solver GenCode and MKLU with RefactChg and RefactVarOpt. TABLE I summarizes the characteristics of each solver. The performances are compared in terms of fill-in reduction, partial refactorization efficiency, pivoting and real-time performance on HIL setup. The applied parallelization techniques are listed in TABLE II.

TABLE I
Compared Solvers

Solvers	Analysis options	Factorization
GenCode	RefactVar (no fill-in reduction)	No pivoting Partial-refactorization
MKLU	AMD, COLAMD or Metis RefactChg or RefactVarOpt	Partial Pivoting, Partial-refactorization or full refactorization according to pivot validity test

TABLE II
Parallelization Techniques

Name	Solution methods
SEQ	Sequential solution of network equations without any decoupling
CM	Compensation Method
LD	Line-Delay method based on the propagation delay of power transmission lines
LD+CM	Combination of Line-Delay and Compensation Method decoupling

A. Fill-in reduction performance

Fill-in reduction is tested on two large linear distribution networks.

Case-1 is the Xavier distribution network [8], with 619 nodes. The simulation interval is 1 s with a time-step of 50 µs, for studying a single-phase-to-ground fault.

Case-2 is the GHOST microgrid case from [25] with 663 nodes. The simulation interval is 90 s with a time-step of 100 µs to simulate a grid fault that provokes an islanded mode.

For these two networks, the main computation effort is on the solution part (backward and forward substitution). Very few refactorizations are required for each case. As there are no natural propagation delay lines for parallel decoupling, the CM is used to accelerate the simulation. TABLE III presents the maximum sizes of subnetwork admittance matrices before/after CM decoupling for each test case. Figures 2 and 3

show the CM decoupling locations for each test case.

TABLE III
ADMITTANCE MATRIX SIZES BEFORE AND AFTER CM DECOUPLING

Case	Parallel solver	Number of tasks	Max Size
Xavier	SEQ	1	619
	CM	2	318
	CM	4	186
GHOST	SEQ	1	663
	CM	5	238

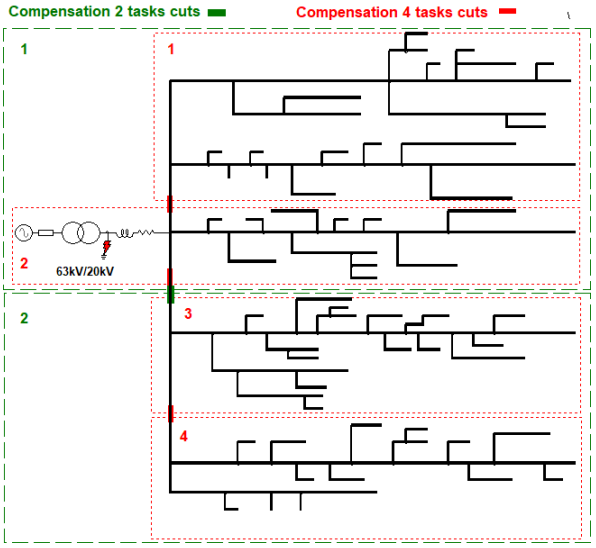

Fig. 2. Xavier distribution test case with CM parallelization.

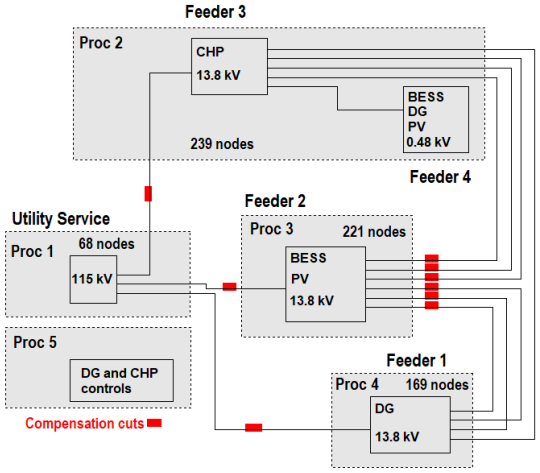

Fig. 3. GHOST microgrid test case with CM parallelization.

Three fill-in techniques are tested with the MKLU solver: AMD, COLAMD and Metis. Figures 4 and 5 display the sparsity patterns of the admittance matrix of each test case and the number of non-zero elements (nz) when using AMD. A lesser non-zero value dispersion from AMD ordering will reduce the fill-in in LU factors.

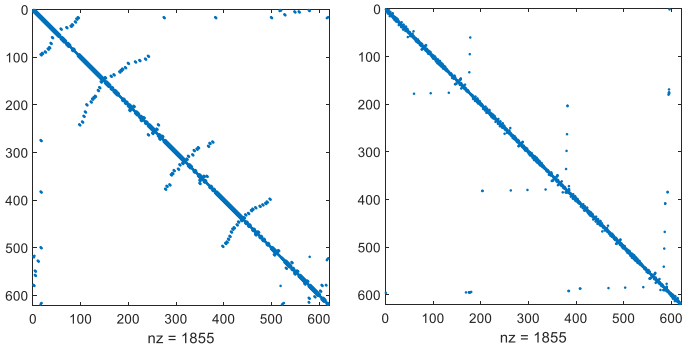

Fig. 4. Sparsity structures before (left) and after (right) AMD ordering for Xavier distribution network.

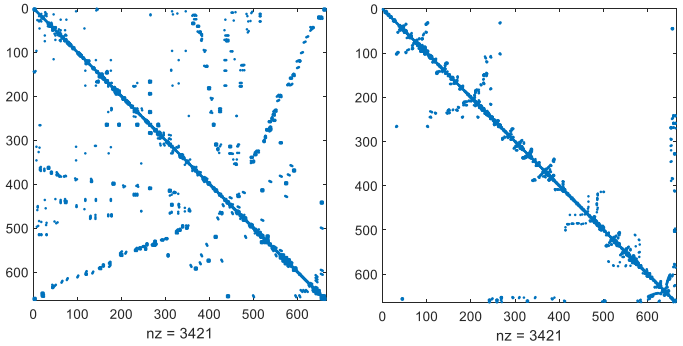

Fig. 5. Sparsity structures before (left) and after (right) AMD ordering for GHOST microgrid.

RefactChg is chosen for the refactorization strategy. Offline and real-time simulations are run on an OP5031 target 64 bits Linux with 32 cores (2 CPU Intel Xeon E5 3.2 GHz – 16 cores). Tables IV and V display performance results for each fill-in reduction technique. The offline average time-step ($\overline{\Delta t}$) is equal to the measured execution time over the total number of time-steps. The related speed-up ratio ($\bar{S}$) is computed against the GenCode solution. For the real-time simulation, no physical hardware is interfaced to the simulator, but the real-time constraint is ensured. The second-last column displays the minimum time-step ($\overline{\Delta t}_{RT}$) to avoid continuous overruns during the real-time simulation. The last column presents the related speed-up ratio ($\bar{S}_{RT}$).

TABLE IV
PERFORMANCE RESULTS, XAVIER DISTRIBUTION NETWORK

Parallelization	Solver	$\overline{\Delta t}$	$\bar{S}$	$\overline{\Delta t}_{RT}$	$\bar{S}_{RT}$
SEQ	GenCode	117.1 µs	1	121 µs	1
	MKLU +AMD	91.7 µs	1.28	96 µs	1.26
	MKLU+COLAMD	91.5 µs	1.28	97 µs	1.25
	MKLU+Metis	92.8 µs	1.26	97 µs	1.25
CM 2 tasks	GenCode	61.4 µs	1	64 µs	1
	MKLU +AMD	52.1 µs	1.18	57 µs	1.12
	MKLU+COLAMD	50.6 µs	1.21	57 µs	1.12
	MKLU+Metis	51.6 µs	1.19	56 µs	1.14
CM 4 tasks	GenCode	35.3 µs	1	37 µs	1
	MKLU +AMD	32.7 µs	1.08	35 µs	1.06
	MKLU+COLAMD	33.4 µs	1.06	35 µs	1.06
	MKLU+Metis	33.1 µs	1.07	35 µs	1.06

TABLE V
PERFORMANCE RESULTS, GHOST MICROGRID

Parallelization	Solver	$\overline{\Delta t}$	$\overline{S}$	$\overline{\Delta t}_{RT}$	$\overline{S}_{RT}$
SEQ	GenCode	64.5 µs	1	68 µs	1
	MKLU +AMD	60.4 µs	1.07	63 µs	1.08
	MKLU+COLAMD	59.9 µs	1.08	63 µs	1.08
	MKLU+Metis	59.3 µs	1.09	61 µs	1.12
CM	GenCode	32 µs	1	38 µs	1
	MKLU +AMD	30.7 µs	1.04	36 µs	1.06
	MKLU+COLAMD	30.6 µs	1.05	36 µs	1.06
	MKLU+Metis	31.4 µs	1.02	36 µs	1.06

The presented results demonstrate the efficiency of fill-in reduction techniques. The performance gain can reach 26% in real-time for Xavier network. As expected, efficiency tends to decrease with decreasing network size. The parallelized version of Xavier network, for example, is less impacted with fill-in reduction.

The combination of MKLU fill-in reduction and CM reaches respectively, for case-1 and case-2, speed-up ratios of 3.5 ($\overline{\Delta t}_{RT\,SEQ+GenCode}/\overline{\Delta t}_{RT\,CM\ 4\ tasks+MKLU}$) and 1.9 ($\overline{\Delta t}_{RT\,SEQ+GenCode}/\overline{\Delta t}_{RT\,CM+MKLU}$) over the legacy solution (SEQ+GenCode). All fill-in reduction techniques give approximately the same performance gain as matrix sizes are not huge. Also, for other cases with transmission lines, LD decouples the network each time a transmission line long enough for decoupling is detected. This limits the maximum size of subnetwork admittance matrices. For the rest of the paper, AMD ordering is kept.

B. Partial refactorization

The test case IFA2000, is an HVDC interconnection between France (Les Mandarins) and United Kingdom (Sellindge). It is used here to assert the efficiency of partial refactorization. The modelling presented in [8] is used. Each LCC pole (Line Commutated Converter) is represented by two detailed 6-pulse bridges. Fig. 6 shows the LD (4 cores) and LD+CM (6 cores) parallel decoupling methods. Several refactorizations are required when the link is in operation which comes from repetitive thyristor commutations. The time-step is set to 30 µs. The 10 s starting sequence real-time software-in-the-loop (SIL) simulation is run on an OP5031 target 32 bits Linux with 32 cores (2 CPU Intel Xeon E5 3.2 GHz - 16 cores).

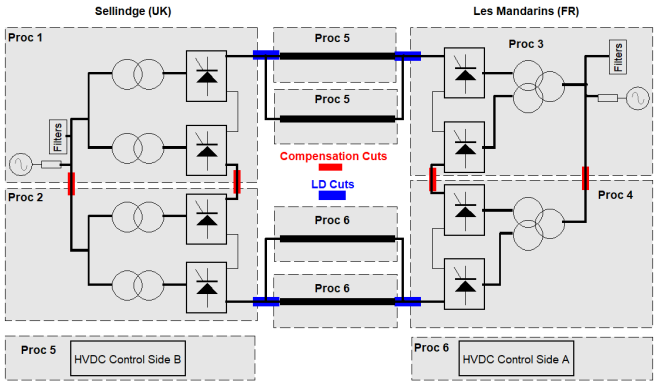

Fig. 6. Overview of IFA2000 modelling and parallel decoupling.

For LD decoupling, the larger subnetwork is Sellindge side. The total size of the admittance matrix is 125x125. The size of the fixed part (linear elements) is 33x33. Fig. 7 shows the sparsity structure of the admittance matrix and fill-in reduction ordering.

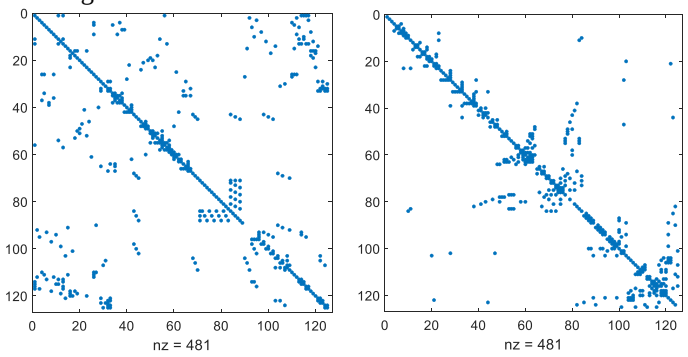

Fig. 7. Sparsity structure before (left) and after (right) AMD ordering for IFA2000 Sellindge subnetwork.

Fig. 9 displays the execution times of the most loaded processor which runs Sellindge side subnetwork. Prior the start of the stations, GenCode is more efficient than MKLU. Fill-in reduction delivers computation gains mainly during the solution phase (backward and forward substitution) as the converter stations have not yet started. This is not enough to over perform the code generation optimization of GenCode. After station start (t>6.4s), fill-in reduction is very effective on the partial factorization stage. MKLU shows performance gains over GenCode. It is even better to applied fill-in reduction to the entire matrix. Indeed, MKLU+RefactChg improves by 50% the performance of GenCode solver. It is the only method with an execution time per time-step far below the 30 µs of the simulation time-step which guarantees a real-time simulation with no overruns. As a counterpart, the solution with this method needs three full refactorizations to cope with pivot invalidity. Applying fill-in reduction in the whole matrix requires pivoting to achieve a stable simulation.

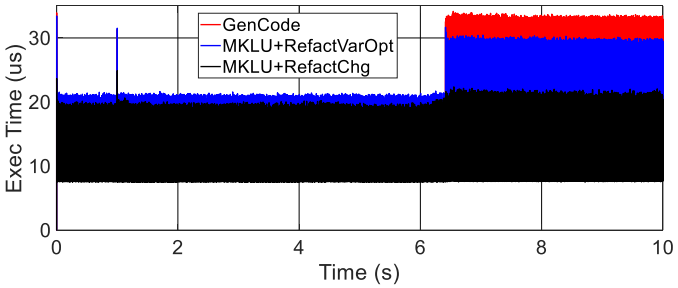

Fig. 8. Execution time of the most loaded processor, LD using various solvers for a 10 s SIL real-time simulation.

With LD+CM decoupling, the maximum matrix size is reduced to 77x77 with a fixed part of 21x21. In that case, as depicted by Fig. 9, the effect of fill-in reduction on performance is less important. RefactVarChg is still more efficient than RefactVarOpt. However, the performances are similar with GenCode. Only 5% of performance gain are obtained when converters are in operation. Code generation optimization avoids less function calls and extra programming structures in comparison with MKLU solver. This counterbalances the fill-in reduction performance gains which are less important with lower matrix dimensions. In that case,

the trade-off between fill-in reduction and further CM parallelization is in favor of fill-in reduction. CM decoupling improves a little over fill-in reduction while using two extra simulation cores.

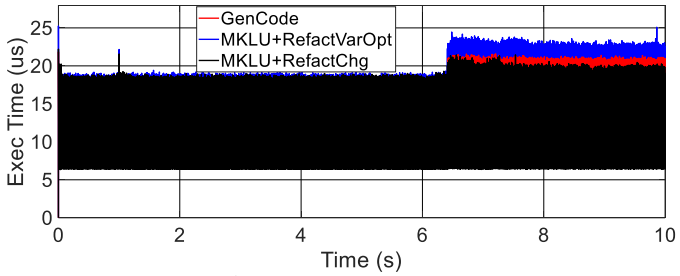

Fig. 9. Execution time of the most loaded processor, LD+CM using various solvers for a 10 s SIL real-time simulation.

C. Pivoting efficiency

The previous part has shown the utility of pivoting when fill-in reduction is applied to the whole matrix. Another example comes from the CM decoupling. It has been already shown in [8][9] that the open circuit solutions for subnetworks may not be stable. The two 6-pulse bridges case (Fig. 10) show this kind of instability. CM decouples each 6-pulse bridge into two subnetworks (Task 1 and Task 2). With this decoupling, the simulation is unstable with GenCode. However, using MKLU helps to achieve a stable simulation. Several pivot changes are required during the computations. MKLU benefits from pivoting to strengthen the stability of CM decoupling. Test case data can be found in the Appendix.

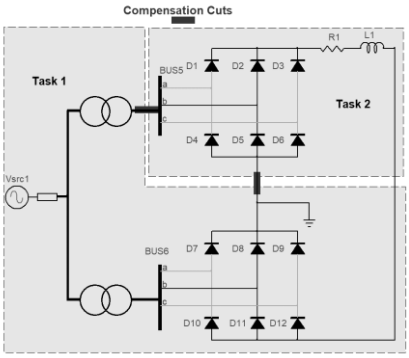

Fig. 10. CM decoupling of two 6-pulse bridges.

D. HIL real-time performance

For real-time HIL simulation, the test case of IFA2000 is considered with the physical cubicles and protections replicas in the simulation loop. The real-time setup is the same as [26]. As depicted in Fig. 11, LD decoupling is used to parallelize the network solution. The simulation runs in real-time with 40 µs time-step on three cores on an OP5031 target 64 bits Linux with 16 cores (CPU Intel Xeon E5 3.2 GHz – 16 cores). The power transit is set to 1000 MW from Les Mandarins to Sellindge. A single phase-to-ground fault occurs on the Les Mandarins side. It is eliminated after a duration of 40 ms. The maximum admittance matrix size (Sellindge subnetwork simulated on Proc 1, see Fig. 11) is 131x131 with a fixed part of 30x30.

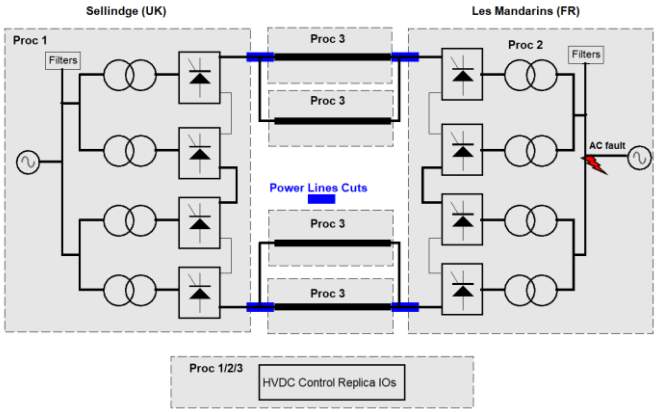

Fig. 11. IFA2000 HIL setup with LD decoupling.

Fig. 12 shows similar performance gains as for the SIL simulation of section III.B. Fill-in reduction applied to the whole matrix (RefactChg) is still the best strategy to save computing times. Pivoting is still necessary for a stable simulation. MKLU+RefactChg reaches a performance gain of 14% over GenCode during HIL real-time simulation. Fig. 13 demonstrates that MKLU gives the same results as GenCode.

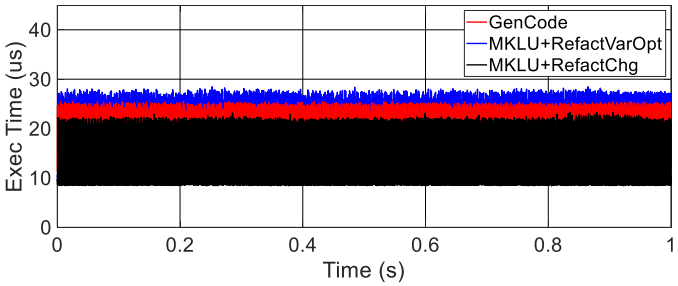

Fig. 12. Execution time of the most loaded processor, LD using various solvers for HIL real-time simulation.

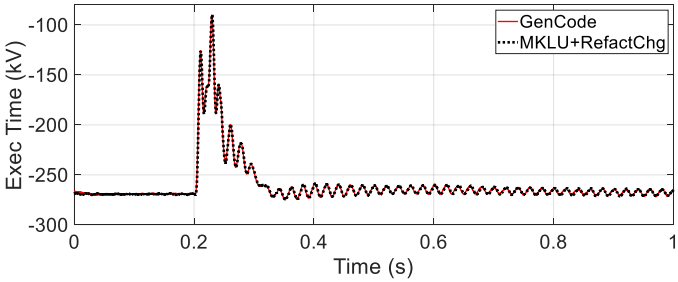

Fig. 13. DC voltage validation during single-phase-to-ground fault.

IV. Conclusion

This paper presented the integration of a direct sparse linear solver, named modified KLU (MKLU), for parallel real-time electromagnetic transient simulations.

First, fill-in reduction techniques embedded in MKLU have provided performance gains of 26% over an existing code generation solver (GenCode) for linear networks. With the compensation method (CM) parallelization technique, fill-in reduction applied to the whole admittance matrix allows reaching a speed-up of 3.9 over the sequential solution with GenCode. For an HVDC network example, the combination with partial refactorization (MKLU+RefactChg) improves by 50% over GenCode when several repetitive refactorizations are required. When the matrix size decreases with further parallel decoupling using LD+CM, fill-in reduction techniques

are less effective.

Second, MKLU offers pivoting which is required to have fill-in reduction over the whole matrix combined with partial refactorization (RefactChg). Without this technique, some simulation cases can become unstable. Additionally, changing the pivot improves the robustness of CM decoupling.

Lastly, a performance gain of 14% has been observed for MKLU+RefactChg for HIL real-time simulation.

Further research is needed to improve the efficiency of partial refactorization.

V. Appendix

Data of the test case presented in Fig. 10:

- For the voltage source, $V_{src} = 86.6\,kV$, $R_{src} = 1.5$, $L_{src} = 0.043\,mH$
- $R_1 = 0.01\,\Omega$, $L_1 = 0.1\,H$
- For each diode, $R_{open} = 10^{-3}\,\Omega$, $R_{close} = 10^{6}\,\Omega$, $R_{snubber} = 100\,\Omega$, $C_{snubber} = 10^{-6}\,\Omega, V_{min} = 0.8\,V$
- For Transformers, Y ground for primary side, Y floating for secondary, $V_{prim} = 86.6\,kV, R_{prim} = 0.43\,\Omega$, $L_{prim} = 0.458\,mH$, $V_{second} = 8.66\,kV, R_{second} = 0.238\,\Omega$, $L_{second} = 0.025\,mH$, for the magnetization branch $R_m = 1.08\,M\Omega$, $L_m = 2.86 \times 10^3\,H$.

VI. References

[1] S. Dennetiere, H. Saad, Y. Vernay, P. Rault, C. Martin and B. Clerc, "Supporting Energy Transition in Transmission Systems: An Operator's Experience Using Electromagnetic Transient Simulation," *IEEE Power Energy Magazine*, vol. 17, no. 3, pp. 48-60, May-June 2019.

[2] H. Saad, P. Rault, S. Dennetière, M. Schudel, C. Wikstrom and K. Sharifabadi, "HIL Simulation to Assess Interaction Risks of HVDC Systems for Upcoming Grid Development," in Proc. *Conf. IEEE Indus. Electron. Soc.* (*IECON)*, Singapore, 2020, pp. 5041-5048.

[3] J. Mahseredjian, S. Dennetière, L. Dubé, B. Khodabakhchian and L. Gérin-Lajoie, "On a new approach for the simulation of transients in power systems," *Elect. Power Syst. Res*, vol. 77, no. 11, 2007, pp.1514-1520.

[4] A. Abusalah, O. Saad, J. Mahseredjian, U. Karaagac and I. Kocar, "Accelerated Sparse Matrix-Based Computation of Electromagnetic Transients," *IEEE Open Access Journal of Power and Energy*, vol. 7, pp. 13-21, 2020.

[5] V. Q. Do, J.-C. Soumagne, G. Sybille, G. Turmel, P. Giroux, G. Cloutier and S. Poulin, "Hypersim, an integrated real-time simulator for power networks and control systems," in *Proc. ICDS'99*, Vasteras, Sweden, May 1999, pp. 1-6.

[6] W. F. Tinney, "Compensation Methods for Network Solutions by Optimally Ordered Triangular Factorization," *IEEE Trans. Power App. Syst.*, vol. PAS-91, no. 1, pp. 123-127, Jan. 1972.

[7] O. Alsac, B. Stott and W. F. Tinney, "Sparsity-Oriented Compensation Methods for Modified Network Solutions," *IEEE Trans. Power App. Syst.*, Vol. PAS-102, no. 5, pp. 1050-1060, May 1983.

[8] B. Bruned, S. Dennetière, J. Michel, M. Schudel, J. Mahseredjian and N. Bracikowski, "Compensation Method for parallel real-time EMT studies," *Elect. Power Syst. Res.*, vol. 198, Sep. 2021.

[9] B. Bruned, J. Mahseredjian, S. Dennetière, J. Michel, M. Schudel and N. Bracikowski, "Compensation Method for Parallel and Iterative Real-Time Simulation of Electromagnetic Transients," *IEEE Trans. Power Del.*, 2023, https://doi.org/10.1109/TPWRD.2023.3238422.

[10] I. S. Duff, A. M. Erisman, J. K. Reid, "Direct Methods for Sparse Matrices Second Edition," *Oxford University Press*, New York, 2017.

[11] T. A. Davis and E. P. Natarajan, "Algorithm 907: KLU, a direct sparse solver for circuit simulation problems," *ACM Trans. Math. Soft.*, vol. 37, no. 3, pp. 36:1-36:17, Sep. 2010.

[12] E. P. Natarajan, "KLU A high performance sparse linear solver for circuit simulation problems," M.S. thesis, Univ. Florida, Gainesville, FL, USA, 2005.

[13] J. Mahseredjian, I. Kocar, U. Karaagac, "Solution Techniques for Electromagnetic Transients in Power Systems", In Transient Analysis of Power Systems: Solution Techniques, Tools and Applications (pp. 9-38), Wiley, 2014, https://doi.org/10.1002/9781118694190.ch2.

[14] Y. Saad, "Iterative Methods for Sparse Linear Systems, Second edition," *SIAM*, Philadelphia, PA, 2003.

[15] S. C. Eisenstat, M. C. Gursky, M. H. Schultz, and A. H. Sherman, *Yale sparse matrix package, I: The symmetric codes*, Internat. J. Nurner. Methods Engrg, 18 (1982), pp. 1145-1151. (Cited on pp. 66, 131.).

[16] A. George and J. W. H. Liu, "The Evolution of the Minimum Degree Ordering Algorithm," *SIAM Review*, vol. 31, no. 1, pp. 1-19, Mar. 1989.

[17] P. R. Amestoy, T. A. Davis and I. S. Duff, "An Approximate Minimum Degree Ordering Algorithm," *SIAM J. Matrix Analysis & Applic*., vol. 17, no. 4, pp. 886-905, Dec.1996.

[18] T. A. Davis, J. R. Gilbert, S. I. Larimore, Esmond G. Ng, "A Column approximate minimum degree ordering algorithm," *ACM Trans. Math. Software*, vol. 30, no. 3, pp. 381-376, Sept. 2004.

[19] G. Karypis and V. Kumar, "Multilevel k-way partitioning scheme for irregular graphs," *J. Parallel Distrib. Comput.*, vol. 48, no. 1, pp. 96–129, 1998.

[20] F. Pellegrini, J. Roman, "SCOTCH: A Software Package for Static Mapping by Dual Recursive Bipartitioning of Process and Architecture Graphs," in *Proc. HPCN'96*, Brussels, Belgium, April 15-19, 1996.

[21] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, Jack J. Dongarra, J. Du Croz, S. Hammarling, A. Greenbaum, A. McKenney, and D. Sorensen, "LAPACK Users' guide (third ed.)," Society for Industrial and Applied Mathematics, USA, 1999.

[22] S. M. Chan and V. Brandwajn, "Partial Matrix Refactorization," *IEEE Trans. Power Syst.*, vol. 1, no. 1, pp. 193-199, Feb. 1986.

[23] J. Dinkelbach, L. Schumacher, L. Razik, A. Benigni, and A. Monti, "Factorisation Path Based Refactorisation for High-Performance LU Decomposition in Real-Time Power System Simulation," *Energies*, vol. 14, no. 23: 7989, Nov. 2021.

[24] B. Bruned, I. M. Martins, P. Rault and S. Dennetière, "Use of Efficient Task Allocation Algorithm for Parallel Real-Time EMT Simulation," *Elect. Power Syst. Res.*, vol. 189, Dec. 2020.

[25] R. Salcedo et al., "Banshee distribution network benchmark and prototyping platform for hardware-in-the-loop integration of microgrid and device controllers," in *The Journal of Engineering*, vol. 2019, no. 8, pp. 5365-5373, Aug. 2019.

[26] Y. Vernay, A. Drouet D'Aubigny, Z. Benalla and S. Dennetière, "New HVDC LCC replica platform to improve the study and maintenance of the IFA2000 link," in *Proc. Int. Conf. Power Syst. Transients (IPST)*, Seoul, Republic of Korea, 2017, pp. 1-6.