Dynamic Simulations of Combined Transmission and Distribution Systems using Parallel Processing Techniques

Petros Aristidou Thierry Van Cutsem

***Abstract*—Simulating a power system with both transmission and distribution networks modeled in detail is a huge computational challenge. In this paper, we propose a Schur-complement-based domain decomposition algorithm to provide accurate, detailed dynamic simulations of the combined system. The simulation procedure is accelerated with the use of parallel programming techniques, taking advantage of the parallelization opportunities inherent in domain decomposition algorithms. The proposed algorithm is general, portable and scalable on inexpensive, shared-memory, multi-core machines. A large-scale test system is used for its performance evaluation.**

***Index Terms*—time simulations, domain decomposition methods, parallel computing, OpenMP**

I. Introduction

The most noticeable developments foreseen in power systems involve Distribution Networks (DNs). Future DNs are expected to host a big percentage of the renewable energy sources. The resulting challenge in dynamic simulation is to correctly represent DNs and their participation in system dynamics. This becomes compulsory as DNs are called upon to actively support the Transmission Network (TN) with an increasing number of Distributed Generation Units (DGUs) and loads participating in ancillary services through smart-grid technologies.

In present-day dynamic security assessment of a large-scale power system, it is common to represent the bulk generation and higher voltage (transmission) levels accurately, while the lower voltage (distribution) levels are equivalenced. On the other hand, when concentrating on a DN, the TN is often represented by a Thévenin equivalent. The prime motivation behind this practice has been the lack of computational resources. Indeed, fully representing the entire power system network was historically impossible given the available computing equipment (memory capacity, processing speed, etc.) [1]. Even with current computational resources, handling the entire, detailed model with hundreds of thousands of Differential and Algebraic Equations (DAE) is extremely challenging [2], [1].

As modern DNs are evolving with power-electronics interfaces, DGUs, active loads, and control schemes, more detailed and elaborate equivalent models would be needed to encompass the dynamics of DNs and their impact on global system dynamics. The three main equivalencing approaches reported in the literature are modal methods, coherency methods and measurement or simulation-based methods [3]. Equivalent models, however, inadvertently suffer from a number of drawbacks:

- the identity of the replaced system is lost. Faults that happen inside the DNs themselves cannot be simulated and individual voltages at internal buses, currents, controllers, etc. cannot be observed anymore;
- most equivalent models target a specific type of dynamics (short-term, long-term, electromechanical oscillations, voltage recovery, etc.) and fail when used for another type. This requires running different types of simulations with different models;
- in most cases, the use or not of these equivalent models is decided off-line, when it is still unknown whether the disturbance will affect or not the DNs of concern.

In this paper, a Schur-complement-based domain decomposition algorithm for the dynamic simulation of combined transmission and distribution systems is presented. The algorithm decomposes the combined system on the boundary between the TN and the DNs. Following, a Schur-complement-based solution is performed to solve the full, detailed DAE system in a decomposed manner.

The proposed algorithm accelerates the simulation procedure in two ways. First, the independent calculations of the sub-networks (such as formulation of non-linear DAE system, discretization, formulation and solution of linear systems, check of convergence, etc.) are parallelized, thus providing computational acceleration. Second, it performs a selective, infrequent Jacobian update, that is, it exploits the decomposition of the system to selectively update only the Jacobian matrices of sub-networks converging more slowly.

The proposed algorithm is parallelized with the use of shared-memory parallel computing techniques through the OpenMP Application Programming Interface (API) targeting common, inexpensive multi-core machines. The implementation is general, with no hand-crafted optimizations particular to the computer system, operating system, simulated electric power network or disturbance.

The paper is organized as follows. In Section II the proposed Schur-complement-based algorithm is presented. In Section III, the parallel processing techniques considered are summarized. Simulation results are reported in Section IV and followed by closing remarks in Section V.

Petros Aristidou is with the Dept. of Elec. Eng. and Comp. Science, University of Liège, Liège, Belgium, e-mail: *p.aristidou@ieee.org*

Thierry Van Cutsem is with the Fund for Scientific Research (FNRS) at Dept. of Elec. Eng. and Comp. Science, University of Liège, Liège, Belgium, e-mail: *t.vancutsem@ulg.ac.be*

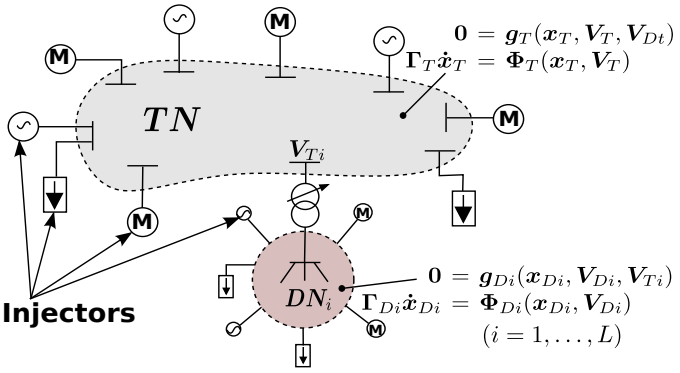

Figure 1. Decomposed Power System

II. Schur-complement-based Algorithm

A. Model Decomposition

An important step in developing and applying a domain decomposition algorithm is the identification of the partition scheme to be used. In this paper, a topologically-based partitioning has been chosen lying on the boundary between TN and DNs. The decomposition assumes that every DN is connected to a single TN bus through a transformer [4].

Let the power system sketched in Fig. 1 be decomposed into the TN and L DNs, along with the power system components connected to them. For reasons of simplicity, all the components connected to the TN or DNs that either produce or consume power in normal operating conditions (such as power plants, DGUs, induction motors, other loads, etc.) are called *injectors*.

The injectors' model can be described by a system of non-linear DAEs [5]:

$$\boldsymbol{\Gamma}\dot{\boldsymbol{x}} = \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{V})$$

where $\boldsymbol{V}$ is the vector of rectangular components of bus voltages ($\boldsymbol{V}_{Di}$ if connected to the i-th DN or $\boldsymbol{V}_T$ if connected to the TN), $\boldsymbol{x}$ is the state vector containing differential and algebraic variables and $\boldsymbol{\Gamma}$ is a diagonal matrix with:

$$(\boldsymbol{\Gamma})_{\ell\ell} = \begin{cases} 0 & \text{if the } \ell\text{-th equation is algebraic} \\ 1 & \text{if the } \ell\text{-th equation is differential.} \end{cases}$$

At the same time, under the standard phasor approximation, the algebraic equations of each network (TN or DN) take on the linear form:

$$\mathbf{0} = \boldsymbol{D}\boldsymbol{V} - \boldsymbol{I} \triangleq \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{V})$$

where $\boldsymbol{D}$ includes the real and imaginary parts of the bus admittance matrix and $\boldsymbol{I}$ is the vector of rectangular components of the bus currents.

Hence, the DAE system describing the TN with its injectors is:

$$\begin{aligned} \mathbf{0} &= \boldsymbol{g}_T(\boldsymbol{x}_T, \boldsymbol{V}_T, \boldsymbol{V}_{Dt}) \\ \boldsymbol{\Gamma}_T\dot{\boldsymbol{x}}_T &= \boldsymbol{\Phi}_T(\boldsymbol{x}_T, \boldsymbol{V}_T) \end{aligned} \tag{1}$$

where $\boldsymbol{V}_{Dt}$ is a sub-vector of $\boldsymbol{V}_D = [\boldsymbol{V}_{D1} \ \ldots \ \boldsymbol{V}_{DL}]^T$, including only the voltage components of the DN buses connected to the TN through the distribution transformers (see Fig. 1).

Similarly, for the i-th DN with its injectors ($i = 1, \ldots, L$):

$$\begin{aligned} \mathbf{0} &= \boldsymbol{g}_{Di}(\boldsymbol{x}_{Di}, \boldsymbol{V}_{Di}, \boldsymbol{V}_{Ti}) \\ \boldsymbol{\Gamma}_{Di}\dot{\boldsymbol{x}}_{Di} &= \boldsymbol{\Phi}_{Di}(\boldsymbol{x}_{Di}, \boldsymbol{V}_{Di}) \end{aligned} \tag{2}$$

where $\boldsymbol{V}_{Ti}$ is a sub-vector of $\boldsymbol{V}_T$, including only the voltage components of the TN bus where the i-th DN is connected to (see Fig. 1).

The proposed decomposition is reflected on the DAE systems (1) and (2), through the common variables $\boldsymbol{V}_{Dt}$ and $\boldsymbol{V}_{Ti}$ ($i = 1, ..., L$) involved in the equations of the distribution transformers connecting the DNs to the TN.

B. Discretization and Algebraization

For the purpose of numerical simulation, the injector DAE systems are discretized using a differentiation formula (such as Trapezoidal Rule, Backward Differentiation Formula, etc.) which yields the corresponding non-linear algebraized system:

$$\mathbf{0} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{V})$$

Next, these injector equations are linearized and solved together with the network equations using a Newton-type method to compute the state vectors $\boldsymbol{V}_T$, $\boldsymbol{x}_T$, $\boldsymbol{V}_{Di}$ and $\boldsymbol{x}_{Di}$.

Thus, at each Newton iteration, the linear system to be solved for the TN is:

$$\begin{aligned} \underbrace{\begin{bmatrix} \boldsymbol{J}_{T1} & \boldsymbol{J}_{T2} \\ \boldsymbol{J}_{T3} & \boldsymbol{J}_{T4} \end{bmatrix}}_{\boldsymbol{J}_T} \begin{bmatrix} \Delta\boldsymbol{V}_T \\ \Delta\boldsymbol{x}_T \end{bmatrix} &- \sum_{i=1}^{L} \begin{bmatrix} \boldsymbol{C}_{Di}\triangle\boldsymbol{V}_{Di} \\ \mathbf{0} \end{bmatrix} = \\ &- \begin{bmatrix} \boldsymbol{g}_T(\boldsymbol{x}_T, \boldsymbol{V}_T, \boldsymbol{V}_{Dt}) \\ \boldsymbol{f}_T(\boldsymbol{x}_T, V_T) \end{bmatrix} \end{aligned} \tag{3}$$

where $\boldsymbol{J}_T$ is the Jacobian matrix of $\boldsymbol{g}_T$ and $\boldsymbol{f}_T$ towards the TN states and $\boldsymbol{C}_{Di}$ towards the voltage of the i-th DN. It is worth noting that the $\boldsymbol{C}_{Di}$ matrix is very sparse with the only non-zero columns corresponding to voltage variables of DN buses directly connected to the TN through the transformers.

Similarly, for the i-th DN, it is:

$$\begin{aligned} \underbrace{\begin{bmatrix} \boldsymbol{J}_{Di1} & \boldsymbol{J}_{Di2} \\ \boldsymbol{J}_{Di3} & \boldsymbol{J}_{Di4} \end{bmatrix}}_{\boldsymbol{J}_{Di}} \begin{bmatrix} \Delta\boldsymbol{V}_{Di} \\ \Delta\boldsymbol{x}_{Di} \end{bmatrix} &- \begin{bmatrix} \boldsymbol{B}_{Di}\triangle\boldsymbol{V}_T \\ \mathbf{0} \end{bmatrix} = \\ &- \begin{bmatrix} \boldsymbol{g}_{Di}(\boldsymbol{x}_{Di}, \boldsymbol{V}_{Di}, \boldsymbol{V}_{Ti}) \\ \boldsymbol{f}_{Di}(\boldsymbol{x}_{Di}, \boldsymbol{V}_{Di}) \end{bmatrix} \end{aligned} \tag{4}$$

where $\boldsymbol{J}_{Di}$ is the Jacobian matrix of $\boldsymbol{g}_{Di}$ and $\boldsymbol{f}_{Di}$ towards the DN_i states and $\boldsymbol{B}_{Di}$ towards the TN bus voltage variables. It can be seen that the $\boldsymbol{B}_{Di}$ matrix is very sparse with the only non-zero columns corresponding to $\boldsymbol{V}_{Ti}$ variables.

C. Reduced System Formulation

The solution of the systems (3)-(4) is performed in a decomposed manner using a Schur-complement-based method. For this, the systems (4) are each solved towards $\Delta\boldsymbol{V}_{Di}$ and substituted in (3) to build a reduced system involving only the TN variables.

To solve for $\Delta\boldsymbol{V}_{Di}$, the system (4) can be rewritten as:

$$\begin{aligned} \boldsymbol{J}_{Di1}\Delta\boldsymbol{V}_{Di} + \boldsymbol{J}_{Di2}\Delta\boldsymbol{x}_{Di} &= -\boldsymbol{g}_{Di}(\boldsymbol{x}_{Di}, \boldsymbol{V}_{Di}, \boldsymbol{V}_{Ti}) \\ &\quad + \boldsymbol{B}_{Di}\triangle\boldsymbol{V}_T \\ \boldsymbol{J}_{Di3}\Delta\boldsymbol{V}_{Di} + \boldsymbol{J}_{Di4}\Delta\boldsymbol{x}_{Di} &= -\boldsymbol{f}_{Di}(\boldsymbol{x}_{Di}, \boldsymbol{V}_{Di}) \end{aligned}$$

$$\begin{bmatrix} \boldsymbol{J}_{T1} - \sum_{i=1}^{L} \boldsymbol{C}_{Di}\tilde{\boldsymbol{B}}_{Di} & \boldsymbol{J}_{T2} \\ \boldsymbol{J}_{T3} & \boldsymbol{J}_{T4} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{V}_T \\ \Delta \boldsymbol{x}_T \end{bmatrix} = - \begin{bmatrix} \boldsymbol{g}_T(\boldsymbol{x}_T, \boldsymbol{V}_T, \boldsymbol{V}_{Dt}) + \sum_{i=1}^{L} \boldsymbol{C}_{Di}\tilde{\boldsymbol{g}}_{Di}(\boldsymbol{x}_{Di}, \boldsymbol{V}_{Di}, \boldsymbol{V}_{Ti}) \\ \boldsymbol{f}_T(\boldsymbol{x}_T, V_T) \end{bmatrix} \quad (5)$$

and then solved for $\Delta \boldsymbol{V}_{Di}$ as:

$$\begin{aligned} \Delta \boldsymbol{V}_{Di} &= + \boldsymbol{S}_{Di}^{-1}\boldsymbol{B}_{Di}\triangle \boldsymbol{V}_T - \boldsymbol{S}_{Di}^{-1}\big[\boldsymbol{g}_{Di}(\boldsymbol{x}_{Di}, \boldsymbol{V}_{Di}, \boldsymbol{V}_{Ti}) \\ &\quad - \boldsymbol{J}_{Di2}\boldsymbol{J}_{Di4}^{-1}\boldsymbol{f}_{Di}(\boldsymbol{x}_{Di}, \boldsymbol{V}_{Di})\big] \\ &= + \tilde{\boldsymbol{B}}_{Di}\triangle \boldsymbol{V}_T - \tilde{\boldsymbol{g}}_{Di}(\boldsymbol{x}_{Di}, \boldsymbol{V}_{Di}, \boldsymbol{V}_{Ti}) \end{aligned}$$

where:

$$\begin{aligned} \boldsymbol{S}_{Di} &\triangleq \boldsymbol{J}_{Di1} - \boldsymbol{J}_{Di2}\boldsymbol{J}_{Di4}^{-1}\boldsymbol{J}_{Di3} \\ \tilde{\boldsymbol{B}}_{Di} &\triangleq \boldsymbol{S}_{Di}^{-1}\boldsymbol{B}_{Di} \\ \tilde{\boldsymbol{g}}_{Di}(\boldsymbol{x}_{Di}, \boldsymbol{V}_{Di}, \boldsymbol{V}_{Ti}) &\triangleq \boldsymbol{S}_{Di}^{-1}\big[\boldsymbol{g}_{Di}(\boldsymbol{x}_{Di}, \boldsymbol{V}_{Di}, \boldsymbol{V}_{Ti}) \\ &\quad - \boldsymbol{J}_{Di2}\boldsymbol{J}_{Di4}^{-1}\boldsymbol{f}_{Di}(\boldsymbol{x}_{Di}, \boldsymbol{V}_{Di})\big] \end{aligned}$$

The resulting equations are then substituted in (3) to formulate the reduced system (5).

It should be noted that the Schur-complement terms $\boldsymbol{C}_{Di}\tilde{\boldsymbol{B}}_{Di}$ each contribute a $[2 \times 2]$ matrix centered on the diagonal of $\boldsymbol{J}_{T1}$. Thus the original sparsity pattern of the TN Jacobian matrix $\boldsymbol{J}_T$ is preserved. Also, each Right-Hand-Side (RHS) factor $\boldsymbol{C}_{Di}\tilde{\boldsymbol{g}}_{Di}(\boldsymbol{x}_{Di}, \boldsymbol{V}_{Di}, \boldsymbol{V}_{Ti})$ affects only the mismatch values of the TN bus where the i-th DN is connected, i.e. only two components when Cartesian coordinates are used.

Finally, all the inverse matrix operations appearing in the above mathematical formulation are actually implemented as sparse linear system solutions, with appropriate solvers, to preserve computational efficiency.

D. Solution

The solution proceeds by solving the reduced system (5) to compute the corrections related to the TN. Then, $\Delta \boldsymbol{V}_T$ and the updated $\boldsymbol{V}_T$ variables are back substituted in Eqs. (4), which are solved to compute the DN corrections. After updating the state vectors, if all the DAE systems (1)-(2) have been solved, the simulation proceeds to the next time instant, otherwise a new iteration is performed with the updated variables.

The solution algorithm performs a "dishonest" update of the Jacobian matrices. That is, the Jacobian matrices $\boldsymbol{J}_T$ and $\boldsymbol{J}_{Di}$, as well as the Schur-complement terms $\boldsymbol{C}_{Di}\tilde{\boldsymbol{B}}_{Di}$ and the intermediate matrices (e.g. $\boldsymbol{S}_{Di}$), are kept constant over several solutions or even time-steps. They are selectively and independently updated only if the corresponding sub-network DAE does not converge after a number of iterations within the same discrete time computation.

The proposed algorithm is numerically equivalent to solving the original DAE system (1)-(2) using a simultaneous Very DisHonest Newton (VDHN) method [6]. The Schur-complement-based algorithm, though, allows to selectively update the Jacobian matrices of DAE sub-systems when needed, and to exploit the decomposition to parallelize the procedure.

III. Parallel Computing Aspects

Domain decomposition-based algorithms offer parallelization opportunities as independent computations can be performed by different computing threads. The proposed algorithm, sketched in Fig. 2, employs parallel computing for the system formulation, Jacobian update and DN solution.

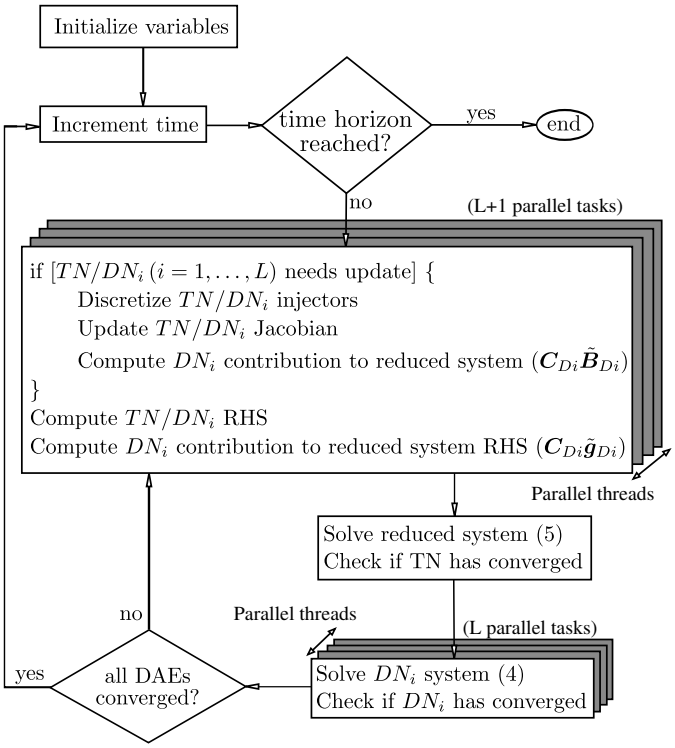

Figure 2. Proposed Solution Algorithm

A. Parallel Algorithm

First, based on the proposed decomposition, no data dependencies exist in the system formulation of each sub-network (TN or DN). Thus, the independent calculations (injector discretization and linearization, Jacobian matrix update, mismatch and reduced system contribution evaluation) are each performed in parallel for the various sub-networks. This is shown in the upper shaded block in Fig. 2 where each parallel task deals with one sub-network. If the $L + 1$ parallel tasks are more than the number of available computational threads, a sharing mechanism takes care of properly assigning the tasks to the threads.

Next, the reduced system (5) is solved to compute the updated TN variables and the convergence of the TN system is checked. Schur-complement-based algorithms suffer from the sequentiality introduced by the reduced system solution [7]. However, due to the high sparsity (retained even after the Schur-complement reduction), the linear nature of the network equations and the infrequent Jacobian update, this bottleneck is bounded to 1-2% of the overall computational cost in the proposed algorithm. Thus, even though this sequentiality could

be tackled with a parallel sparse linear solver, the overhead due to the new synchronization points would counteract the benefits. Hence, in this work, a sequential sparse linear solver has been used.

Finally, after the computed corrections $\Delta \boldsymbol{V}_T$ are back substituted in Eqs. (4), the DN systems are decoupled, removing any data dependencies. The solution of DN sub-systems is obtained in parallel, using sparse linear solvers, and their convergence is checked. This is shown in the lower shaded block in Fig. 2 where each parallel task deals with one DN.

B. Implementation Specifics

Shared-memory, multi-core computers are becoming more and more popular among low-end and high-end users due to their availability, variety and performance at low prices. The OpenMP API was selected for this implementation as it is supported by most hardware and software vendors and it allows for portable, user-friendly programming.

OpenMP has the major advantage of being widely adopted, thus allowing the execution of a parallel application, without changes, on many different computers. It consists of a set of compiler directives, library routines, and environment variables that influence run-time behavior. A set of predefined directives are inserted in Fortran, C or C++ programs to describe how the work is to be shared among threads that will execute on different processors or cores and to order accesses to shared data [8].

One of the most important tasks is to make sure that parallel threads receive equal amounts of work. Imbalanced load sharing leads to delays, as some threads are still working while others have finished and remain idle. OpenMP includes three easy to employ mechanisms for achieving good load balance among the working threads [8].

First, with the *static* strategy, the scheduling is predefined and one or more parallel tasks are assigned to each thread rotationally prior to the parallel execution. This decreases the overhead needed for scheduling but can introduce load imbalance if the work inside each task is not the same. Second, with the *dynamic* strategy, the scheduling is updated during the execution. This introduces a high overhead cost for managing the threads but provides the best possible load balancing. Finally, with the *guided* strategy, the scheduling is again dynamic but the number of tasks assigned to each thread are progressively reduced in size. This way, scheduling overheads are reduced at the beginning of the loop and good load balancing is achieved at the end.

In the proposed algorithm, high imbalance between parallel tasks can arise from the different sizes of the various sub-networks (TN or DNs). That is, if the sub-networks in the system have different number of buses and injectors, hence different size of DAE systems, the threads computing them will have different work loads. In such situations, the dynamic strategy is preferred for better load balancing. Spatial locality can be addressed by defining a minimum number of successive tasks to be assigned to each thread (*chunk*). Temporal locality, on the other hand, cannot be easily addressed with this strategy because the tasks treated by each thread, and thus the data accessed, are decided at run-time and can change from one parallel segment of the code to the next [8].

IV. Results

In this section we present the results of the Schur-complement-based algorithm implemented in the academic simulation software RAMSES, developed at the University of Liège. The software is written in modern Fortran 2003 with the use of OpenMP directives for the parallelization as detailed in Section III. The simulations are performed on a 48-core AMD Opteron Interlagos[1] desktop computer running Debian Linux 6. The environment variable *OMP_NUM_THREADS* was used to vary the number of computational threads available to the simulation software at each execution.

A. Performance Indices

Many different indices exist for assessing the performance of a parallel algorithm $\mathcal{A}$. The two indices used in this study, *scalability* and *speedup*, are defined as [9]:

$$Scalability(N) = \frac{Wall\ time\,(\mathcal{A})\ (1\,core)}{Wall\ time\,(\mathcal{A})\ (N\,cores)} \tag{6}$$

$$Speedup(N) = \frac{Wall\ time\,(VDHN)\ (1\,core)}{Wall\ time\,(\mathcal{A})\ (N\,cores)} \tag{7}$$

where N is the number of available computational threads.

The first index shows how the parallel implementation scales when the number of available processors increases. That is, the tested parallel algorithm is benchmarked against a sequential execution of the *same* algorithm.

The scalability index is directly related to Amdahl's law [8] and using the latter, can be rewritten as:

$$Scalability(N) = \frac{S+P}{S+\frac{P}{N}+OHC(N)} \tag{8}$$

where S is the sequentially computed portion, P the parallel portion and OHC the OverHead Cost of making the code run in parallel (creating and managing threads, communication, memory latency, etc.). The values of S and P can be estimated with the use of a profiler monitoring the sequential execution of the algorithm. Equations (6) and (8) can be used to assess the algorithm's parallel efficiency, defined as the net incremental acceleration gained with each additional computational thread.

Usually, parallel algorithms are designed and optimized to be executed in parallel and exhibit low performance in sequential execution. Thus, even though scalability is an important index, it is not enough to assess the absolute performance of a parallel algorithm. Hence, the speedup index (7) shows how much faster is the proposed parallel algorithm compared to a fast, optimized for sequential execution, algorithm. In this study, the sequential VDHN algorithm was used as a reference. In this algorithm, the combined DAE system (1)-(2)

[1]CPU 6238 @ 2.60GHz, 16KB private L1, 2048KB shared per two cores L2 and 6144KB shared per six cores L3 cache, 128GB RAM

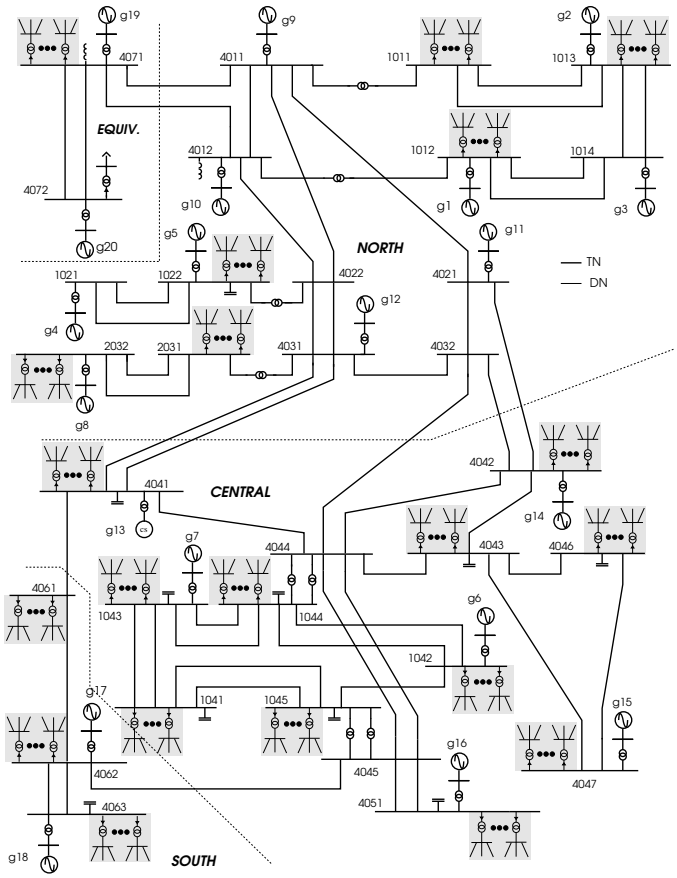

Figure 3. Expanded Nordic System

is solved as a whole using a Newton method with infrequent Jacobian update. The full DAE system is discretized and linearized, the combined Jacobian matrix is formulated and the linear system solved to compute all variable corrections simultaneously. The Jacobian matrix is updated only if the solution has not converged after three iterations. This is a well-known method used by many commercial and academic software and considered to be one of the fastest sequential algorithms [9].

It is noteworthy that both algorithms were implemented in the same software. Thus, they solve exactly the same model equations to the same accuracy, using the same algebraization method (namely the second-order backward differentiation formula), way of handling the discrete events [10], mathematical libraries (i.e. sparse linear solver), etc. Keeping the aforementioned parameters the same for both algorithms permits for the better evaluation of the proposed algorithm's performance.

B. Test System Model

This section reports on results obtained with a large-scale combined transmission and distribution network model based on the Nordic system, documented in [11]. The original TN model is extended with 146 realistic DNs replacing its aggregated distribution loads as shown in Fig. 3. The model and data of each DN were taken from [12] and scaled to match

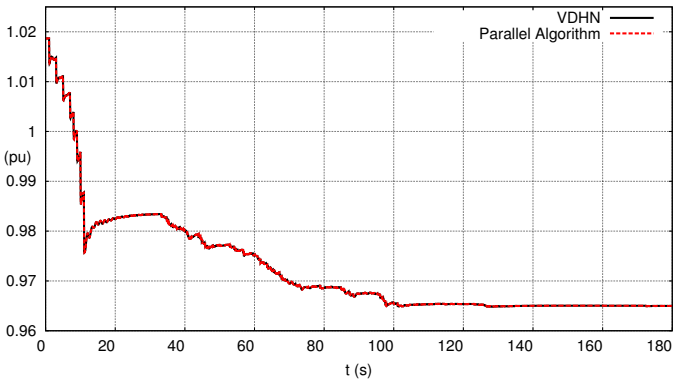

Figure 4. Case 1: Voltage on TN bus 1041

the original TN loads. Multiple DNs were used to match the original load powers, taking into account the nominal power of the TN-DN transformers.

Each one of the 146 DNs includes 100 buses, 108 branches, one distribution voltage regulator equipped with Load Tap Changing (LTC) device, three type-1, three type-2 and two type-3 Wind Turbines (WTs) [13], 12 impedance loads and 133 dynamically modeled loads, such as small induction machines and self-restoring exponential loads. The transformer connecting each DN to the TN is also equipped with an LTC controlling the distribution side voltage.

To further avoid identical DNs and artificial synchronization, the delays on transformer tap changes were randomized around their original values and the WTs were randomly initialized to produce 80-100% of their nominal power.

In total, the combined transmission and distribution system includes 14653 buses, 15994 branches, 20 synchronous machines, 293 LTC equipped transformers, 1168 WTs, 1752 impedance loads and 19419 dynamically modeled loads. The resulting DAE system has 143462 differential-algebraic states.

C. Case 1

The disturbance considered in this scenario is the loss of approximately 115 MW of wind generation due to the disconnection of 30 type-1, 30 type-2 and 20 type-3 WTs located inside ten DNs, all connected to TN bus 1041 in the CENTRAL area (see Fig. 3). The WTs, grouped per DN, are successively disconnected over a period of 10 s and the system is simulated for 180 s with a time-step size of one cycle at the nominal frequency (50 Hz). This event might result from high winds in the area, causing WTs to trip to avoid damage. Such disturbances, with events happening inside the DNs, are very difficult to simulate when detailed DN models are not used.

Figure 4 shows the voltage at the TN bus 1041, where the affected DNs are attached. The successive disconnection of the WTs inside the DNs is reflected on the voltage evolution during the first 10 s. That is, as the WTs within each DN disconnect, the corresponding DN imports the lost power from the TN. This gradually increasing TN-DN power transfer leads to depressed TN voltages. In the long term, the system evolves under the effect of LTCs acting to restore distribution

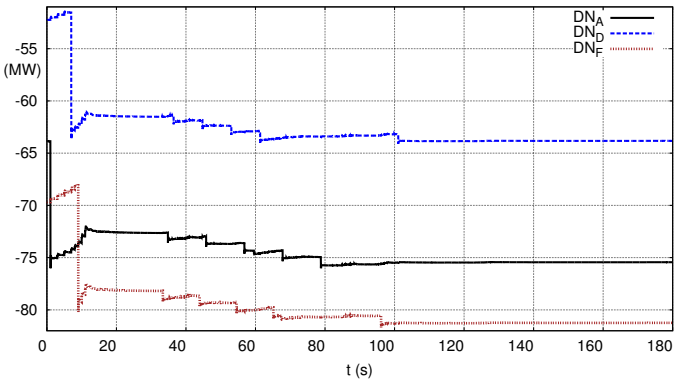

Figure 5. Case 1: Active power transfer over TN-DN transformers (negative sign signifies the DN is importing active power from the TN)

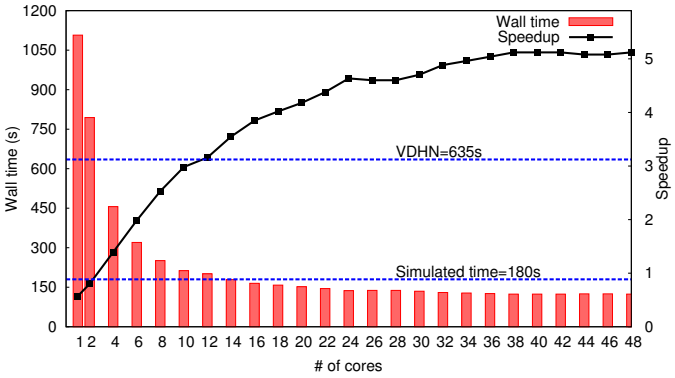

Figure 6. Case 1: Speedup

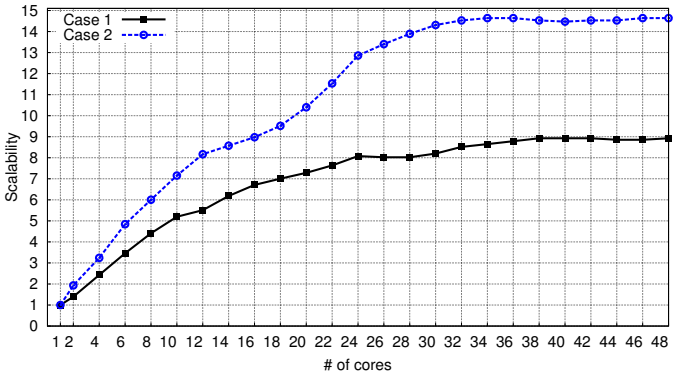

Figure 7. Case 1 and 2: Scalability

voltages, thus further depressing TN voltages. The simulated evolution is shown with both VDHN and the proposed parallel algorithm. As expected, the output trajectories are indistinguishable as the two algorithms solve the same DAEs with the same accuracy.

Figure 5 shows the active power transfer over the TN-DN transformers of three of the DNs. In particular, DN_A is the first, DN_D is the fourth and DN_F the sixth whose WTs disconnect. When the WTs of a DN disconnect, the imported power is immediately increased to compensate for the lost local generation. The already mentioned TN voltage drop impacts the neighboring DNs and, due to the voltage sensitivity of loads, the imported power decreases. Hence, immediately after the disturbance, only a fraction of the lost WT power is imported from the TN. In the long-term, as the LTC actions restore the DN voltages, the load consumption is also restored and the whole lost active power deficit is compensated by importing from the TN.

This interaction mechanism shows the necessity for detailed DN representation in dynamic simulations. The sequence of discrete events, like WT disconnections, LTC actions, etc., the behavior of DN components and controls, and the interactions of DNs with the TN or between them, dictate the resulting system evolution.

Figure 6 shows that algorithm $\mathcal{A}$ offers a speedup of up to 5.2 times when compared to the VDHN algorithm. Initially, the parallel algorithm executed on a single core performs around 40% slower than the, optimized for sequential execution, VDHN. This delay is due to the extra computational costs of the domain decomposition-based scheme (e.g. partition-related book-keeping, intermediate Schur-complement calculations, etc.). As regards the scalability of the algorithm, Fig. 7 shows that it executes up to nine times faster in parallel compared to its own sequential execution.

From Figs. 6 and 7, it can be seen that the parallel algorithm is more efficient in the range of up to 24 cores, while after that the benefit becomes marginal. This can be explained from Eq. (8). When increasing the number of parallel threads by one, the execution time gained can be computed as $\frac{P}{N} - \frac{P}{N+1}$, assuming that the sequential and parallel portions remain unchanged. Thus, when N increases, it is easy to see that the incremental gain decreases. At the same time, the incremental OHC of creating and managing a new thread $(OHC(N+1) - OHC(N))$, calculated for the specific computer platform, is almost constant. Hence, as the number of computational threads increases, the net incremental gain (difference between incremental gain and OHC) declines and can reach zero or even negative values.

D. Case 2

The disturbance considered in this scenario is a five cycle (0.1 s) short circuit near the TN bus 4032 cleared by the opening line 4032-4042. The system is then simulated over 240 s with one cycle time-step size. After the electromechanical oscillations have died out, the system evolves in the long-term under the effect of LTCs acting to restore distribution voltages and overexcitation limiters on the generators. This is a severe disturbance that affects all the TN and DNs.

Figure 8 shows the active power output of a type-3 WT located in one of the DNs of the CENTRAL area. It is shown with both VDHN and the proposed parallel algorithm. As expected, the two are indistinguishable.

Figure 7 shows the scalability of the proposed algorithm reaching 14.5 times, while Fig. 9 shows that the parallel algorithm achieves a speedup of eight times compared to the VDHN, simulating the disturbance in around 130 s.

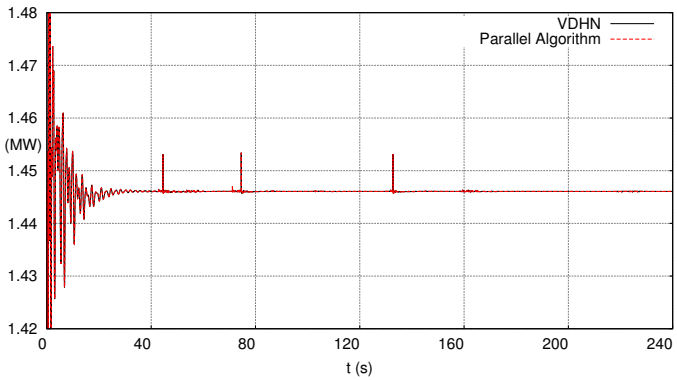

Figure 8. Case 2: DN type-3 WT active power output

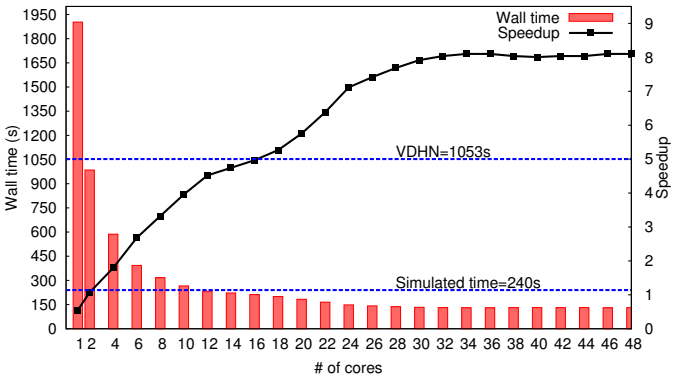

Figure 9. Case 2: Speedup

It can be seen that both scalability and speedup are higher in Case 2 compared to Case 1. Moreover, the algorithm exhibits efficient scaling up to 32 cores, compared to the 24 cores of the previous case. These differences can be explained by the larger amount of computational work available in the parallel portion of this simulation. Indeed, due to the severe nature of this scenario, the system exhibits higher dynamic activity. Thus, more frequent Jacobian matrix updates and more DN system solutions are required, leading to an increased overall computation time ($S + P$). At the same time, as most of the aforementioned computations are in the parallel portion, the ratio $\frac{P}{S+P}$ also increases. Hence, with a bigger P value, the higher scalability and speedup achieved can be explained from Eq. (8), considering that the incremental OHC, that depends on the computer platform, is the same as before.

In general, the proposed algorithm is more efficient and achieves higher speedup when simulations with high dynamic activity are considered.

V. Conclusion

In the future, distributed protection and control schemes, DGUs providing ancillary services and active demand response will make the contribution of DNs to the system dynamics more significant and their detailed simulation more vital. Thus, the need for simulating larger power system models, including DNs, will increase the computational burden of dynamic simulations.

In this paper a parallel Schur-complement-based algorithm for dynamic simulation of combined transmission and distribution systems has been presented. The algorithm yields acceleration of the simulation procedure in two ways. On the one hand, the procedure is accelerated numerically, by performing selective and infrequent Jacobian updates of the decomposed sub-systems. On the other hand, it is accelerated computationally, by exploiting the parallelization opportunities inherent to domain decomposition algorithms.

The proposed algorithm is accurate, as the original system of equations is solved with the same accuracy. It has the ability to simulate a wide variety of disturbances. It exhibits high numerical convergence rate, provided by Newton-type algorithms.

Along with the proposed algorithm, an implementation based on the shared-memory parallel programming model has been presented. The implementation is portable, as it can be executed on any platform supporting the OpenMP API. It can handle general power systems, as no hand-crafted, system specific, optimizations were applied. Finally, it exhibits good parallel performance on inexpensive, shared-memory, multi-core computers.

References

[1] D. Koester, S. Ranka, and G. Fox, "Power systems transient stability-A grand computing challenge," *Northeast Parallel Architectures Center, Syracuse, NY, Tech. Rep. SCCS*, vol. 549, 1992.

[2] R. Green, L. Wang, and M. Alam, "High performance computing for electric power systems: Applications and trends," in *Proc. of IEEE PES General Meeting*, 2011.

[3] U. D. Annakkage, N. K. C. Nair, Y. Liang, A. M. Gole, V. Dinavahi, B. Gustavsen, T. Noda, H. Ghasemi, A. Monti, M. Matar, R. Iravani, and J. A. Martinez, "Dynamic System Equivalents: A Survey of Available Techniques," *IEEE Transactions on Power Delivery*, vol. 27, pp. 411–420, Jan. 2012.

[4] T. Short, *Electric Power Distribution Handbook*. Electric power engineering series, Taylor & Francis, 2003.

[5] P. Kundur, *Power system stability and control*. McGraw-hill New York, 1994.

[6] B. Stott, "Power system dynamic response calculations," *Proceedings of the IEEE*, vol. 67, no. 2, pp. 219–241, 1979.

[7] Y. Saad, *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, 2nd ed., 2003.

[8] B. Chapman, G. Jost, and R. Van Der Pas, *Using OpenMP: Portable Shared Memory Parallel Programming*. MIT Press, 2007.

[9] J. Chai and A. Bose, "Bottlenecks in parallel algorithms for power system stability analysis," *IEEE Transactions on Power Systems*, vol. 8, no. 1, pp. 9–15, 1993.

[10] D. Fabozzi, A. Chieh, P. Panciatici, and T. Van Cutsem, "On simplified handling of state events in time-domain simulation," in *Proc. of the 17th Power Systems Computation Conference*, 2011.

[11] T. Van Cutsem, "Description, modeling and simulation results of a test system for voltage stability analysis," Internal Report, University of Liège, Sept. 2013. http://hdl.handle.net/2268/141234.

[12] A. Ishchenko, *Dynamics and stability of distribution networks with dispersed generation*. PhD thesis, Dept. Electrical. Eng., Univ. TU/E, the Netherlands, 2008.

[13] A. Ellis, Y. Kazachkov, E. Muljadi, P. Pourbeik, and J. Sanchez-Gasca, "Description and technical specifications for generic WTG models: A status report," in *IEEE PES Power Systems Conference and Exposition*, March 2011.