A Parallel Processing Approach to Dynamic Simulations of Combined Transmission and Distribution Systems[☆]

P. Aristidou

Department of Electrical Engineering and Computer Science,
University of Liège, Liège, Belgium

T. Van Cutsem

Fund for Scientific Research (FNRS) at Department of Electrical Engineering and Computer Science, University of Liège, Liège, Belgium

Abstract

Simulating a power system with both transmission and distribution networks modeled in detail is a huge computational challenge. In this paper, a Schur-complement-based domain decomposition algorithm is proposed to provide accurate, detailed dynamic simulations of such systems. The simulation procedure is accelerated with the use of parallel programming techniques, taking advantage of the parallelization opportunities inherent to 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.

Keywords: time simulations, domain decomposition methods, parallel computing, OpenMP, Schur-complement

1. 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

[☆] An earlier version of this paper was presented at the 2014 Power Systems Computation Conference.

Email addresses: p.aristidou@ieee.org (P. Aristidou), t.vancutsem@ulg.ac.be (T. Van Cutsem)

To appear in International Journal of Electrical Power & Energy Systems 2015

(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 large-scale power systems, 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 (DAEs) is extremely challenging [1, 2].

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 the 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. This makes it difficult to simulate controls or protections that rely on these values;
- 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. Moreover it adds an additional burden of maintaining and updating these models when changes are made to the DN;
- in most cases, the use or not of these equivalent models is decided offline, 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 (i.e. desktops or laptops). 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 2 the proposed Schur-complement-based algorithm is presented. In Section 3, the parallel processing techniques considered are summarized. Simulation results are reported in Section 4 and followed by closing remarks in Section 5.

2. Schur-complement-based Algorithm

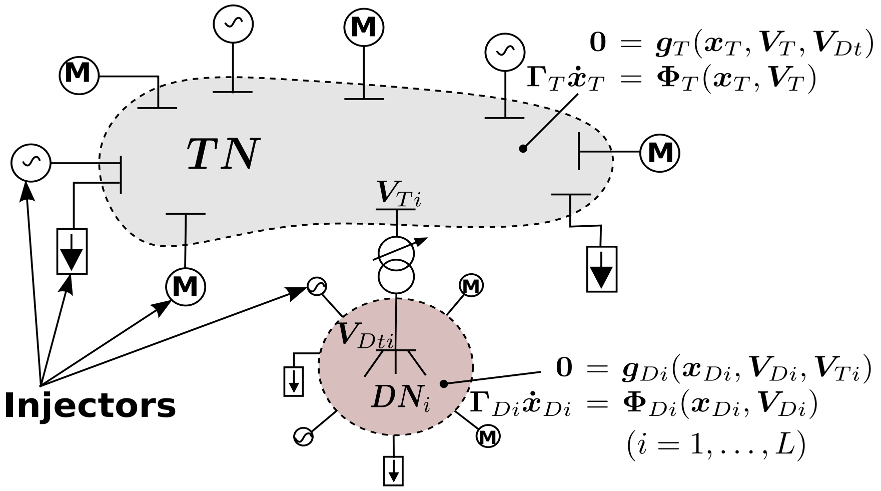
$\mathbf{0} = \boldsymbol{g}_T(\boldsymbol{x}_T, \boldsymbol{V}_T, \boldsymbol{V}_{Dt})$
$\boldsymbol{\Gamma}_T \dot{\boldsymbol{x}}_T = \boldsymbol{\Phi}_T(\boldsymbol{x}_T, \boldsymbol{V}_T)$
$\boldsymbol{TN}$
$\boldsymbol{V}_{Ti}$
$\boldsymbol{V}_{Dti}$
$\boldsymbol{DN}_i$
$\mathbf{0} = \boldsymbol{g}_{Di}(\boldsymbol{x}_{Di}, \boldsymbol{V}_{Di}, \boldsymbol{V}_{Ti})$
$\boldsymbol{\Gamma}_{Di} \dot{\boldsymbol{x}}_{Di} = \boldsymbol{\Phi}_{Di}(\boldsymbol{x}_{Di}, \boldsymbol{V}_{Di})$
$(i = 1, \ldots, L)$
Injectors
M

Figure 1: Decomposed Power System

2.1. 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]:

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

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 $\mathbf{\Gamma}$ is a diagonal matrix with:

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

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:

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

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 ($i = 1, \ldots, L$):

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

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 system through the presence of $\boldsymbol{V}_{Dt}$ in the Eq. 1 of the TN and $\boldsymbol{V}_{Ti}$ ($i = 1, ..., L$) in the Eq. 2 of the i-th DN.

2.2. Discretization and Algebraization

For the purpose of numerical simulation, the injector DAE systems are discretized using an implicit 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:

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

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:

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

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.

2.3. 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 $\Delta\boldsymbol{V}_{Di}$, Eqs. 4 can be rewritten as:

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

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

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

where:

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

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

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

It should be noted that the Schur-complement terms $\boldsymbol{C}_{Di}\tilde{\boldsymbol{B}}_{Di}$ each contribute a [2 × 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.

2.4. 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.

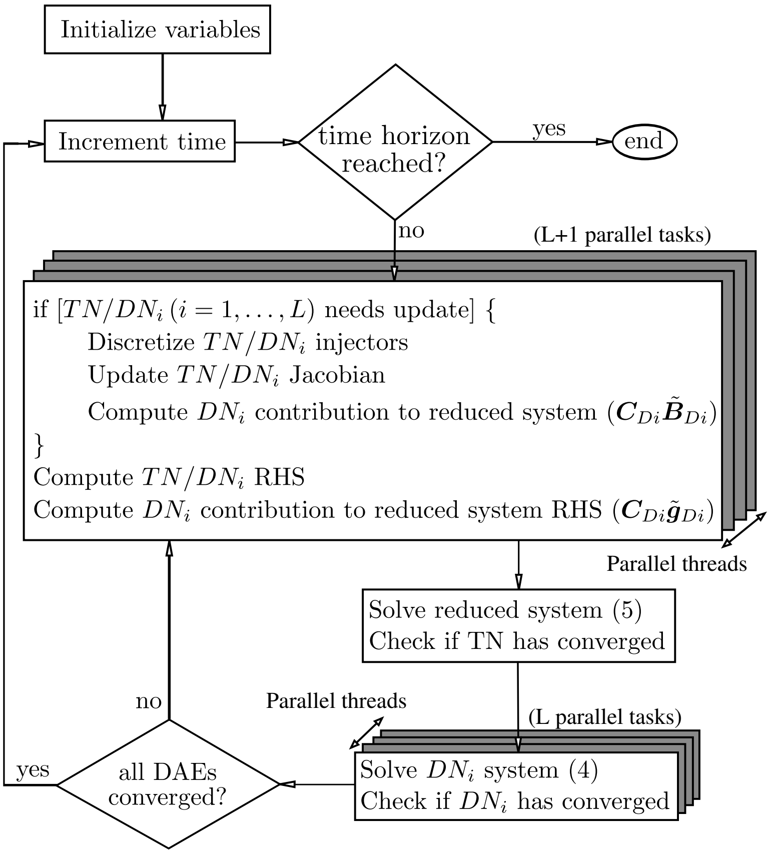
Initialize variables
Increment time
time horizon reached?
yes
end
no
(L+1 parallel tasks)
if $[TN/DN_i\,(i=1,\ldots,L)$ needs update$]$ {
Discretize TN/DN_i injectors
Update TN/DN_i Jacobian
Compute DN_i contribution to reduced system $(\boldsymbol{C}_{Di}\tilde{\boldsymbol{B}}_{Di})$
}
Compute TN/DN_i RHS
Compute DN_i contribution to reduced system RHS $(\boldsymbol{C}_{Di}\tilde{\boldsymbol{g}}_{Di})$
Parallel threads
Solve reduced system (5)
Check if TN has converged
no
Parallel threads
(L parallel tasks)
yes
all DAEs converged?
Solve DN_i system (4)
Check if DN_i has converged

Figure 2: Proposed Solution Algorithm

3. 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.

3.1. 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.

3.2. 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 (namely *static*, *dynamic* and *guided*) for achieving good load balance among the working threads [8].

The static strategy is used mainly when the work within each parallel task is almost the same. Thus, the scheduling is predefined with the parallel tasks assigned to the threads evenly prior to the execution. This strategy has the lowest scheduling overhead among the three but can introduce load imbalance if the work inside each task is not equal. Following, the dynamic strategy is used when the work within each parallel task is highly imbalanced. To cope with that, the scheduling is updated during the execution. This strategy has the highest overhead cost for managing the threads but provides the best possible load balancing. Finally, the guided strategy is a compromise between the other two. The scheduling in this strategy is dynamic but the number of tasks assigned to each thread is 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, there is inherently a high imbalance between parallel tasks due to 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. Consequently, the dynamic strategy has been chosen for better load balancing. Furthermore, by defining a minimum number of successive tasks to be assigned to each thread (*chunks*) and positioning the task data consecutively in memory, spatial locality can be exploited. That is, the likelihood of accessing consecutive blocks of memory is increased and the amount of cache misses decreased [8].

4. Results

In this section the results of the Schur-complement-based algorithm are presented. The algorithm is 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 API directives for the parallelization as detailed in Section 3. The simulations were performed on a 48-core AMD Opteron

Interlagos[1] desktop computer running Debian Linux 6 and the environment variable *OMP_ NUM_ THREADS* was used to vary the number of computational threads at each execution.

4.1. 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) can be used to show 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) 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

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

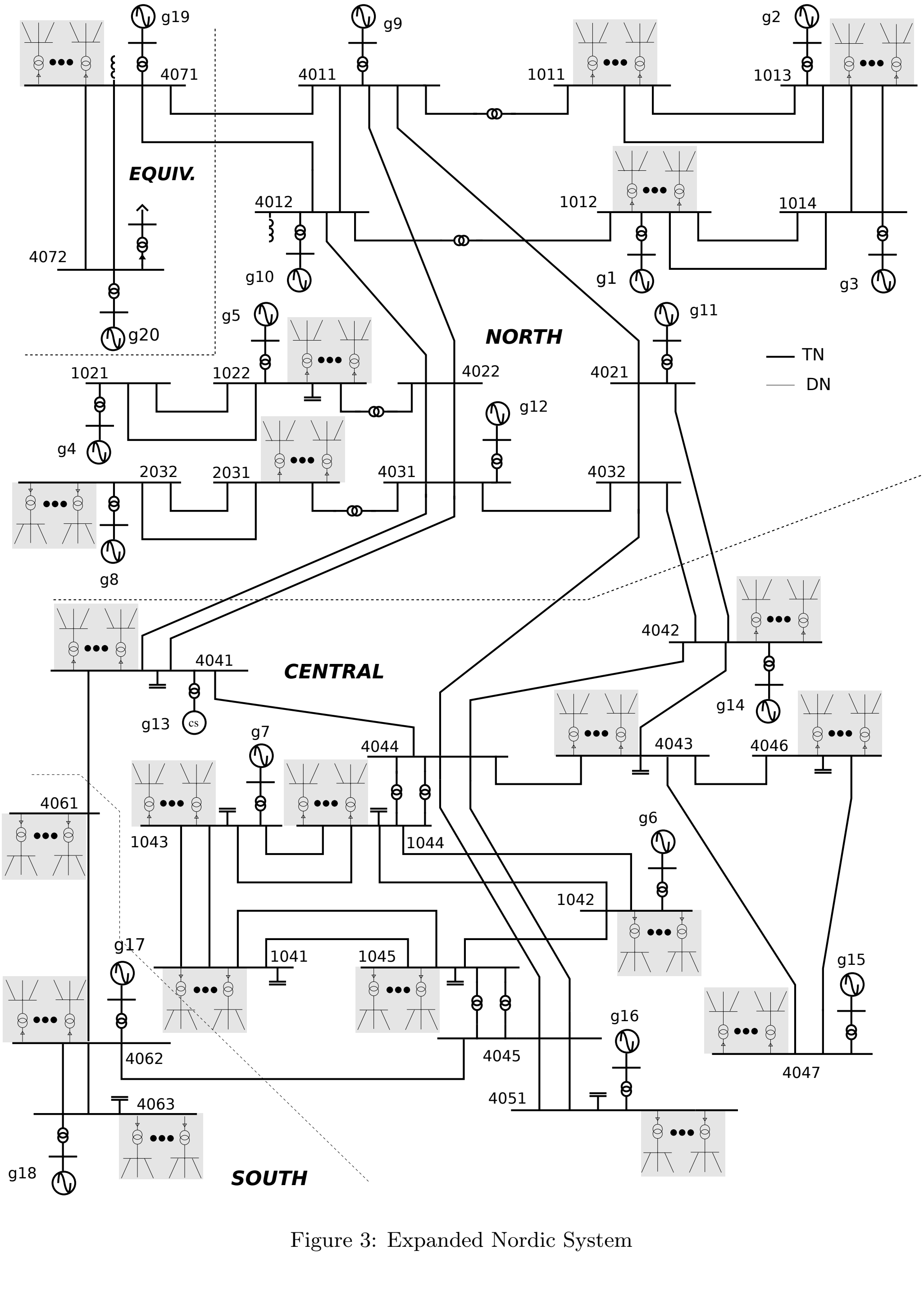

Figure 3: Expanded Nordic System

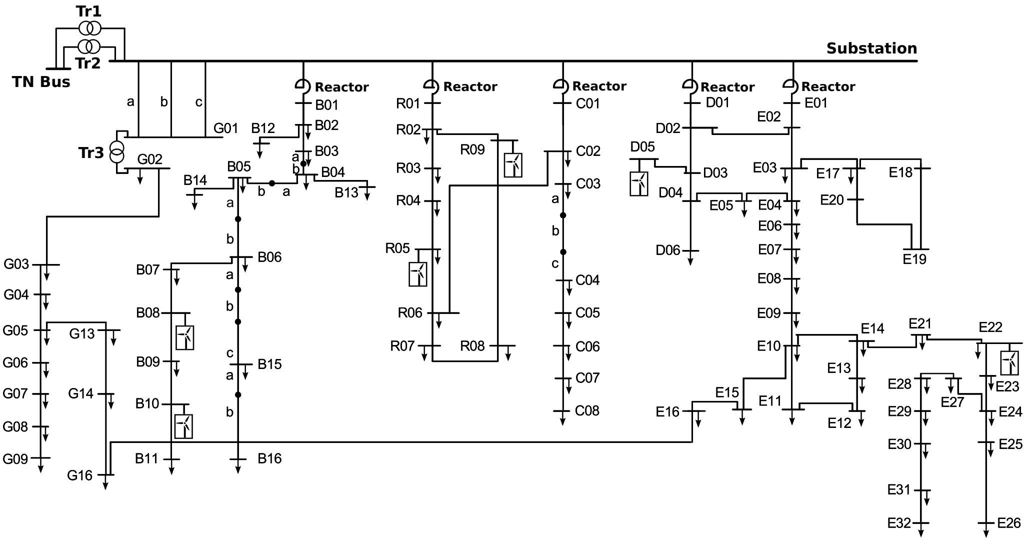

Figure 4: Detailed DN model

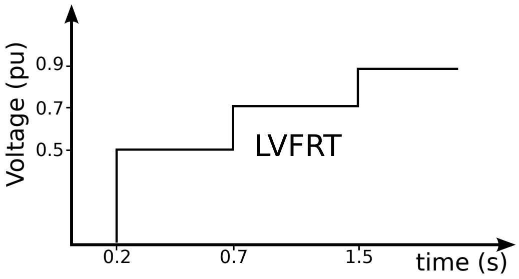

Figure 5: LVFRT capability curve of WTs

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 has to be noted that both algorithms were implemented in the same software. Thus, exactly the same model equations are solved 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. By keeping the aforementioned parameters the same for both algorithms, the evaluation of the proposed algorithm's performance is facilitated.

4.2. 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 TN model (presented in Fig. 3) is expanded with 146 DNs (shown in Fig. 4) that replace the aggregated distribution loads. The model and data of each DN were taken from [12] and scaled to match the original loads seen by the TN. 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 is connected to the TN through two parallel transformers equipped with Load Tap Changing (LTC) devices. Each DN includes 100 buses, one distribution voltage regulator equipped with LTC, six type-2 and two type-3 Wind Turbines (WTs) [13] and 133 dynamically modeled loads, such as small induction machines and exponential loads.

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. Moreover, each WT complies with the Low Voltage and Fault Ride Through (LVFRT) requirements sketched in Fig. 5 as described in [14].

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

4.3. Case 1: Short-term Stability Study

In this scenario a short-term stability study is presented. The disturbance considered is a 8-cycle 3-phase fault near the TN bus 4044 cleared by the opening the faulted line 4043-4044 in the CENTRAL area (see Fig. 3). The system is simulated for 10 s with a time-step size of half cycle at the nominal frequency (50 Hz). The simulation is performed twice: once taking into account the LVFRT characteristics (*Case 1a*) and once keeping all the WTs connected throughout the entire simulation (*Case 1b*).

Figure 6 shows the voltage at some TN buses for Cases 1a and 1b. It can be seen that the system is short-term stable in both cases. However, the final values in Case 1b are higher than in 1a. This can be explained from Fig. 7, depicting the total active power generated by WTs throughout the simulation. The disconnection of the WTs in accordance with the LVFRT curves leads to losing approximately 150 MW of distributed wind generation. This deficit is covered by importing more power from the TN, thus leading to depressed TN voltages.

Furthermore, Fig. 8 shows the voltage at buses E22 and R09 (see Fig. 4) in two different DNs connected on TN buses 1041 and 4043, respectively. These

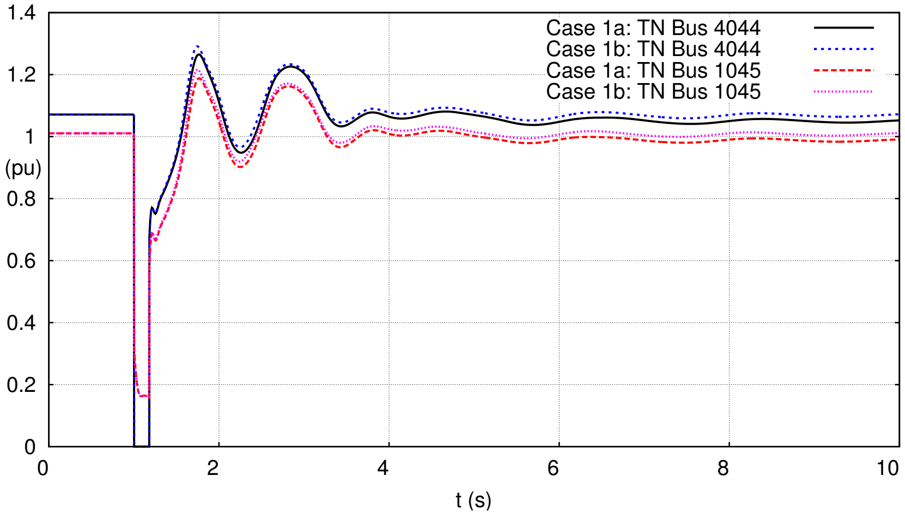

Figure 6: Case 1: Voltages on selected TN buses

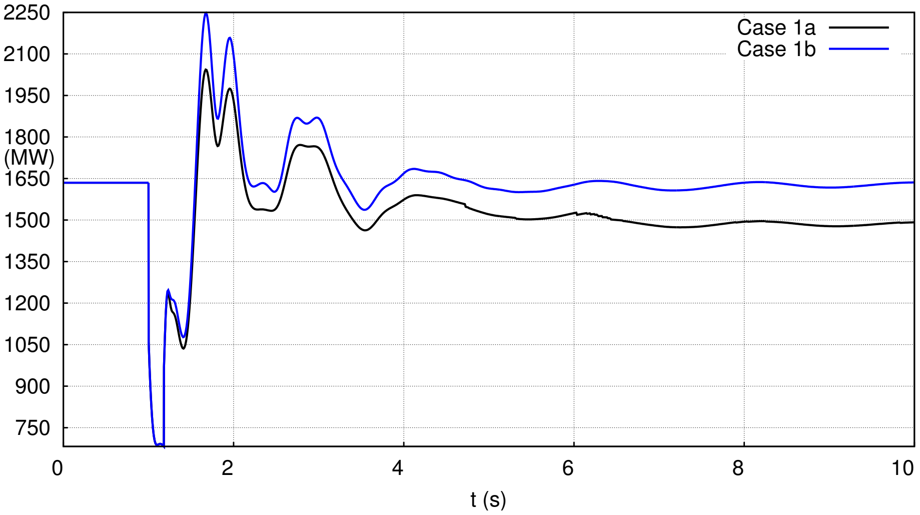

Figure 7: Case 1: Total active power generation by WTs in all the DNs

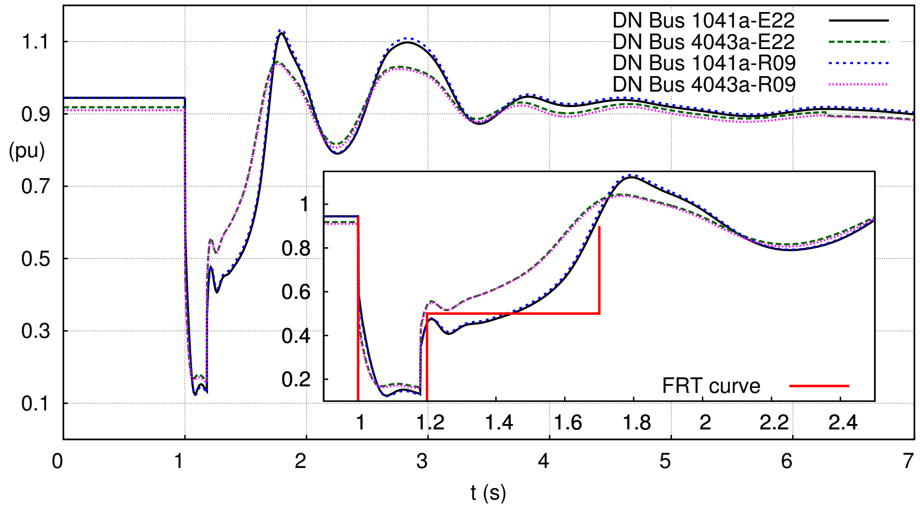

Figure 8: Case 1a: Voltages in DNs

buses have WTs attached to them and the voltage evolution can be compared to the LVFRT curve of Fig. 5 to know whether they will remain connected or not. It can be seen that the WTs in the DN 1041a disconnect at $t \approx 1.2$ s while in DN 4043a, they remain connected. The LVFRT disconnection relies on the voltage evolution at DN buses, illustrating the necessity for detailed combined dynamic simulations.

Figure 9 shows the speedup of the proposed algorithm compared to the VDHN algorithm. Initially, the parallel algorithm executed on a single core performs similarly to the VDHN. When using more computational cores, the proposed algorithm offers a speedup of up to 15 times and the system is simulated in approximately 20 seconds. As regards the scalability of the algorithm, Fig. 10 shows that it executes up to 15 times faster in parallel compared to its own sequential execution.

From Figs. 9 and 10, 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.

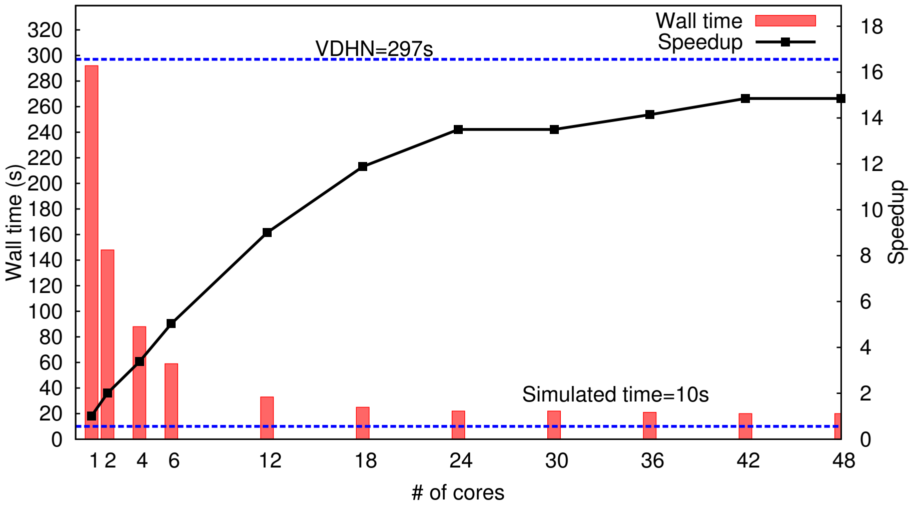

Figure 9: Case 1a: Speedup

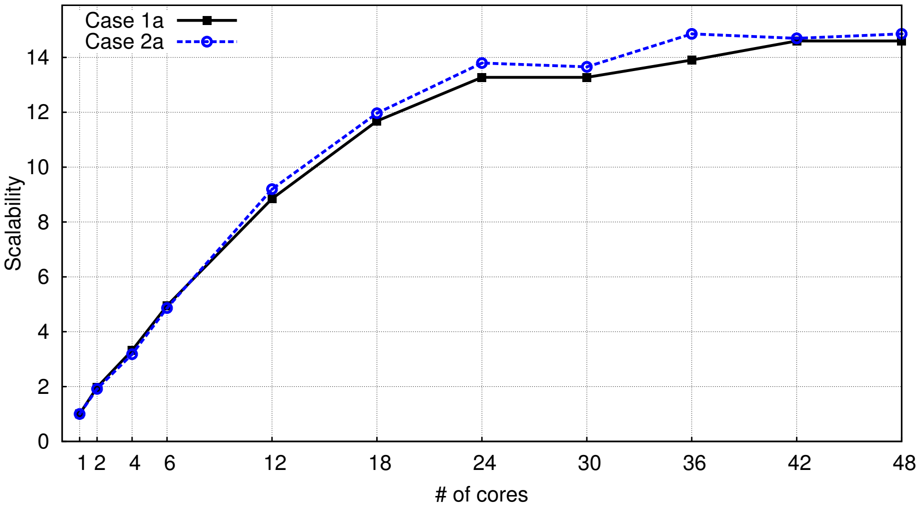

Figure 10: Cases 1a and 2a: Scalability

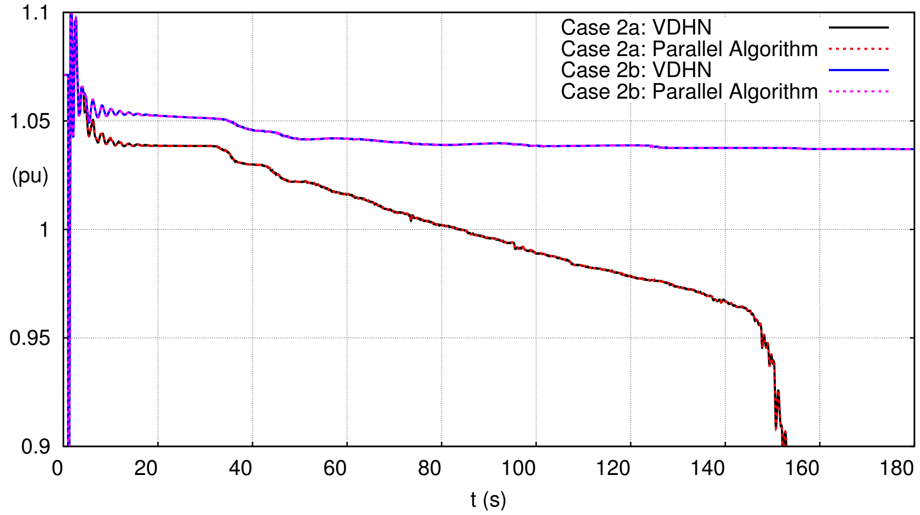

Figure 11: Case 2: Voltage on TN bus 4044

4.4. Case 2: Long-term Stability Study

In this section a long-term stability study is presented. The disturbance considered is a 5-cycle 3-phase fault near the TN bus 4032 cleared by the opening the faulted line 4032-4042. The system is simulated for 180 s with a time-step size of half cycle in the short-term (10 s) and one cycle for the remaining. Here too, the simulation is performed taking into account the LVFRT curves (*Case 2a*) and keeping all the WTs connected throughout the entire simulation (*Case 2b*).

Figure 11 shows the voltage at the TN bus 4044 for Cases 2a and 2b. Both cases are short-term stable. 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. It can be seen that, while Case 2b is long-term stable, Case 2a is unstable with the system collapsing at $t \approx 150$ s.

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.

In Case 2a, the successive disconnection of WTs inside the DNs, in accordance with the LVFRT, is reflected on the voltages. Figure 12 shows the total active power generated by WTs. It can be seen that the WT disconnection leads to losing approximately 140 MW. As the WTs disconnect, the DNs import the lost power from the TN and this increased TN-DN power transfer leads to depressed TN voltages. Moreover, the LTCs act to restore distribution voltages and consequently the consumption of voltage sensitive loads. This leads to a further voltage depression at the TN level until the system collapses.

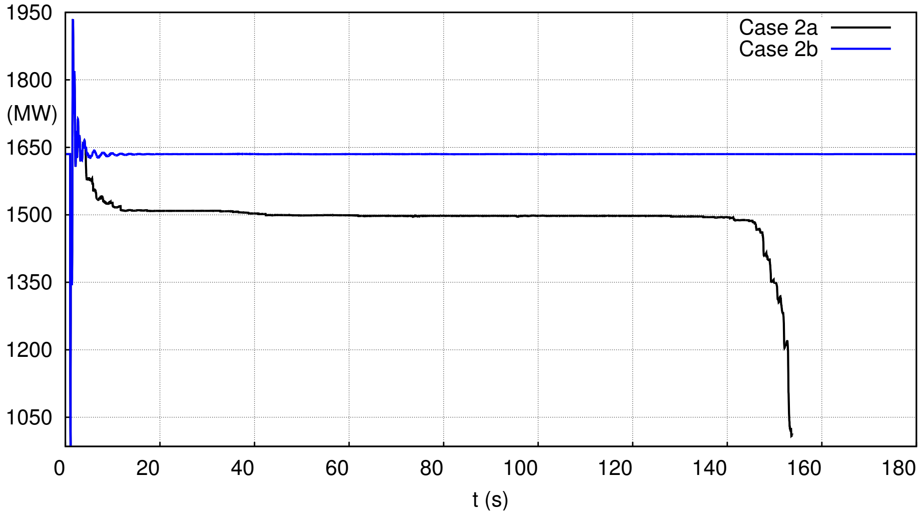

Figure 12: Case 2: Total active power generation by WTs in all the DNs

On the other hand, in Case 2b the WTs remain connected to the DNs throughout the simulation, thus supporting the system; the long-term voltage collapse is avoided. This complex 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 system evolution.

Figure 13 shows that the proposed algorithm offers a speedup of up to 11.5 times compared to the VDHN algorithm. Initially, the parallel algorithm executed on a single core performs around 32% 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. 10 shows that it executes up to 15 times faster in parallel compared to its own sequential execution. The algorithm exhibits efficient scaling up to 24 cores, similarly to Case 1.

It can be seen that while scalability is similar, the speedup is higher in Case 1 compared to Case 2. In Case 1, only the short-term period after the fault is simulated. During this period, the system exhibits high dynamic activity, thus more frequent Jacobian matrix updates and DN solutions are needed leading to an increased overall computation time ($S+P$). At the same time, as most of the aforementioned computations are in the parallel portion (see Fig. 2) the ratio $\frac{P}{S+P}$ also increases. Hence, Eq. 8 can be used to explain the higher speedup, considering that the incremental OHC is practically constant for the given computational

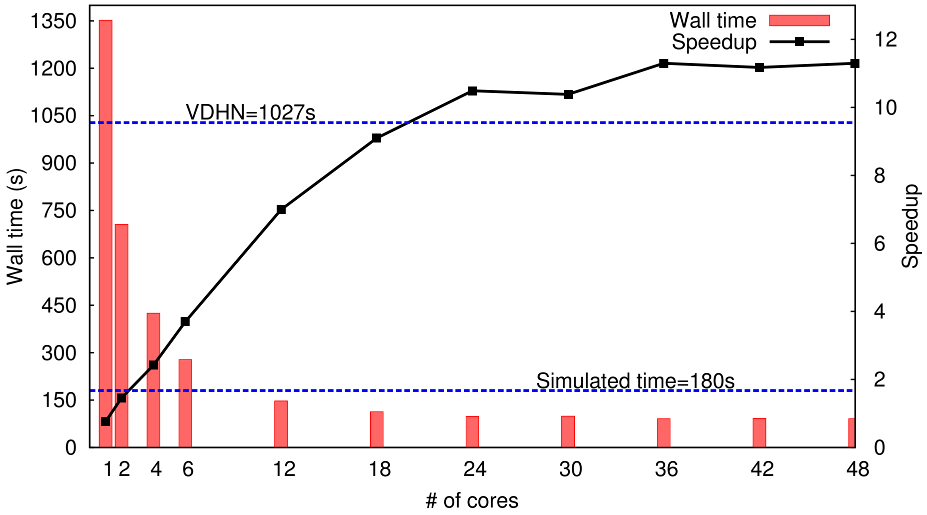

Figure 13: Case 2a: Speedup

platform.

On the other hand, in Case 2, the long-term period is also considered where the Jacobian matrix updates and DN solutions are not so frequent. Hence, the speedup is slightly lower.

5. 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. The need for simulating power system models including DNs, will dramatically 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. Moreover, it exhibits good parallel performance on inexpensive, shared-memory, multi-core computers.

Finally, it was shown that the proposed algorithm is more efficient in simulations with high dynamic activity.

References

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

[2] R. Green, L. Wang, M. Alam, High performance computing for electric power systems: Applications and trends, in: Proc. of the IEEE PES General Meeting, 2011. doi:10.1109/PES.2011.6039420.

[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, J. A. Martinez, Dynamic System Equivalents: A Survey of Available Techniques, IEEE Transactions on Power Delivery 27 (1) (2012) 411–420. doi:10.1109/TPWRD.2011.2167351.

[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 67 (2) (1979) 219–241. doi:10.1109/PROC.1979.11233.

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

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

[9] J. Chai, A. Bose, Bottlenecks in parallel algorithms for power system stability analysis, IEEE Transactions on Power Systems 8 (1) (1993) 9–15. doi:10.1109/59.221242.

[10] D. Fabozzi, A. Chieh, P. Panciatici, T. Van Cutsem, On simplified handling of state events in time-domain simulation, in: Proc. of the 17th Power Systems Computation Conference, 2011.
URL `http://hdl.handle.net/2268/91650`

[11] T. Van Cutsem, L. Papangelis, Description, modeling and simulation results of a test system for voltage stability analysis, Internal Report, University of Liège (Nov. 2013).
URL `http://hdl.handle.net/2268/141234`

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

[13] A. Ellis, Y. Kazachkov, E. Muljadi, P. Pourbeik, J. Sanchez-Gasca, Description and technical specifications for generic WTG models: A status report, in: Proc. of the IEEE PES Power Systems Conference and Exposition, 2011. doi:10.1109/PSCE.2011.5772473.

[14] J. Schlabbach, Low voltage fault ride through criteria for grid connection of wind turbine generators, in: Proc. of the 5th International Conference on European Electricity Market (EEM), 2008. doi:10.1109/EEM.2008.4578999.