# A Schur Complement Method for DAE Systems in Power System Dynamic Simulations

Petros Aristidou[1], Davide Fabozzi[1], and Thierry Van Cutsem[2]

## 1 Introduction

Power system dynamic simulations are widely used in industry and academia to provide important information on the dynamic evolution of a system after the occurrence of a disturbance. In modern dynamic simulation software there is the need to represent complex electric equipment that interact with each other directly or through the network. These equipment models represent generators, motors, loads, wind generators, compensators, etc. with all the physics involved and the required controls. This multi-domain modeling leads to large, non-linear, stiff and hybrid (i.e. subject to both continuous and discrete dynamics) Differential and Algebraic Equation (DAE) systems [10].

In these dynamic simulation studies, the speed of simulation is of the utmost importance. The observations resulting from these simulations can be critical in scheduling corrective actions to guard the actual power system against instability. This procedure, called real-time Dynamic Security Assessment, is performed by many power system companies.

Triggered mainly by the developments in parallel processing technologies, some DDMs have already been proposed to speed up simulations. They are mainly based on Schwartz alternating methods and Waveform Relaxation methods [9, 7, 11]. Unlike space domain decomposition, no geometrical information is given to decompose a DAE system [5] and engineers have to rely on a priori information on the system's topology and operation for partitioning. Furthermore, alternating algorithms demand great care in the partitioning of the system and the handling of interface values to ensure the convergence of the methods [1, 8]. If tightly coupled unknowns are mapped to different partitions and an alternating procedure is used, significantly slowed down convergence rates or divergence can be experienced [2].

This paper proposes a robust, accurate and efficient parallel algorithm based on the direct Schur Complement DDM [13]. The algorithm yields significant acceleration when compared to classic, high performance, integrated (applied on the undecomposed system) dynamic simulation algorithms. The two-fold gain comes from utilizing the parallel potential of the method and exploiting the locality and sparsity of power systems. Furthermore, as a direct method, convergence does not depend on the specific partitioning of the system as the interface values are resolved accurately at each step before solving the sub-domain problems. A connection between the

[1] Department of Electrical Engineering and Computer Science, University of Liège, Liège, Belgium, e-mail: p.aristidou@ieee.org .[2] Fund for Scientific Research (FNRS) at the Department of Electrical Engineering and Computer Science, University of Liège, Liège, Belgium, e-mail: t.vancutsem@ulg.ac.be

proposed algorithm and quasi-Newton based integrated algorithms is demonstrated allowing the better comprehension of the algorithm's properties. Finally, an implementation of the algorithm using the shared-memory parallel programming model and some numerical results are presented based on a realistic large-scale test system.

The paper is organized as follows: in Section 2 we present the partitioning scheme of the proposed algorithm. In Section 3, we explain the formulation of the dynamic simulation problem and the solution using the Schur Complement method. In Section 4, some further investigation of the algorithm is made with the help of quasi-Newton integrated algorithms. Implementation specifics and simulation study are reported in Section 5 and followed by closing remarks in Section 6.

## 2 Power System Modeling

An electric power system, under the phasor approximation [10], can be described in compact form by the following DAE Initial Value Problem:

$$\begin{aligned} \mathbf{0} &= \boldsymbol{\Psi}(\mathbf{x}, \mathbf{V}) \\ \boldsymbol{\Gamma}\dot{\mathbf{x}} &= \boldsymbol{\Phi}(\mathbf{x}, \mathbf{V}) \\ \mathbf{x}(t_0) = \mathbf{x}_0, &\mathbf{V}(t_0) = \mathbf{V}_0 \end{aligned} \tag{1}$$

where $\mathbf{V}$ is the vector of voltages through the network, $\mathbf{x}$ is the expanded state vector containing the differential and algebraic variables (except the voltages) of the system and $\boldsymbol{\Gamma}$ is a diagonal matrix with $\boldsymbol{\Gamma}_{\ell\ell} = \begin{cases} 0, & \text{if the } \ell\text{-th equation is algebraic} \\ 1, & \text{if the } \ell\text{-th equation is differential.} \end{cases}$

The first part of system (1) corresponds to the purely algebraic network equations. The second part describes the remaining DAEs of the system. Discrete events (caused by digital controllers, load tap changing devices, etc.) can alter the power system equations during the simulation. The handling of these discrete events is not presented in this paper [4].

### *2.1 Power System Partitioning*

First, the purely algebraic equations describing the electric network are separated to create one sub-domain. Then, each model of a component connected to the network (such as a synchronous machine, a load, a motor or even a low-voltage distribution network) is separated to form the remaining sub-domains. All the aforementioned devices connected to the network will be called *injectors*. This term encompasses devices that either produce or consume power in normal operating conditions. Each injector is assumed to be connected to a single bus of the network and the interface is on the physical junction between the sub-domains. Extension to two or more connection buses is straightforward [4]. The decomposition is visualized in Fig. 1.

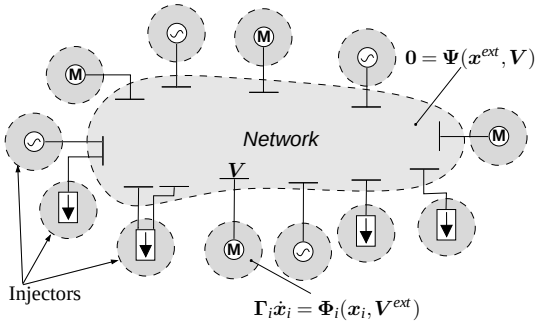



**Fig. 1 Decomposed Power System:** The Power System is decomposed into the Network and the injectors connected to it. This reveals a star shaped decomposition layout with the Network sub-domain connected to all other sub-domains.

The network sub-domain is described by the algebraic equation system (2) while the sub-domain of each injector $i$ is described by the DAE system (3).

$$\begin{aligned} \mathbf{0} &= \boldsymbol{\Psi}(\mathbf{x}^{ext}, \mathbf{V}) \\ \mathbf{x}^{ext}(t_0) &= \mathbf{x}_0^{ext}, \mathbf{V}(t_0) = \mathbf{V}_0 \end{aligned} \tag{2}$$

$$\begin{aligned} \boldsymbol{\Gamma}_i \dot{\mathbf{x}}_i &= \boldsymbol{\Phi}_i(\mathbf{x}_i, \mathbf{V}^{ext}) \\ \mathbf{x}_i(t_0) &= \mathbf{x}_{i0}, \mathbf{V}^{ext}(t_0) = \mathbf{V}_0^{ext} \end{aligned} \tag{3}$$

Sub-domains numbered $1, \ldots, M-1$ relate to injectors and $M$ relates to the network. Vectors $\mathbf{x}_i$ and matrices $\boldsymbol{\Gamma}_i$ are the projections of $\mathbf{x}$ and $\boldsymbol{\Gamma}$, defined in (1), on the $i$ sub-domain. The variables of each sub-domain are separated into interior (*int*) variables appearing only in equations of the sub-domain itself and interface (*ext*) variables appearing in equations of both the Network and an injector sub-domain. Thus, for injectors $\mathbf{x}_i = [\mathbf{x}_i^{int}\ \mathbf{x}_i^{ext}]$ and for the Network $\mathbf{V} = [\mathbf{V}^{int}\ \mathbf{V}^{ext}]$ (see Fig. 1).

## 3 DDM-based Algorithm

### *3.1 Local System Formulation*

Each injector DAE sub-system is algebraized and the resulting non-linear systems of equations are solved with a quasi-Newton method. The local linear systems involved in the solution take on the form of (4) for the injectors and (5) for the network.

$$\underbrace{\begin{bmatrix} \mathbf{A}_{1i}\ \mathbf{A}_{2i} \\ \mathbf{A}_{3i}\ \mathbf{A}_{4i} \end{bmatrix}}_{\mathbf{A}_i} \underbrace{\begin{bmatrix} \triangle \mathbf{x}_i^{int} \\ \triangle \mathbf{x}_i^{ext} \end{bmatrix}}_{\triangle \mathbf{x}_i} + \underbrace{\begin{bmatrix} \mathbf{0}\ \mathbf{B}_i \end{bmatrix}}_{\widetilde{\mathbf{B}}_i} \begin{bmatrix} \mathbf{0} \\ \triangle \mathbf{V}^{ext} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{f}_i^{int}(\mathbf{x}_i^{int}, \mathbf{x}_i^{ext}) \\ \mathbf{f}_i^{ext}(\mathbf{x}_i^{int}, \mathbf{x}_i^{ext}, \mathbf{V}^{ext}) \end{bmatrix}}_{\mathbf{f}_i} \tag{4}$$

$$\underbrace{\begin{bmatrix}\mathbf{D}_1 & \mathbf{D}_2\\ \mathbf{D}_3 & \mathbf{D}_4\end{bmatrix}}_{\mathbf{D}}\underbrace{\begin{bmatrix}\triangle\mathbf{V}^{int}\\ \triangle\mathbf{V}^{ext}\end{bmatrix}}_{\triangle\mathbf{V}}+\sum_{j=1}^{M-1}\underbrace{\begin{bmatrix}\mathbf{0} & \mathbf{C}_j\end{bmatrix}}_{\widetilde{\mathbf{C}}_j}\begin{bmatrix}\mathbf{0}\\ \triangle\mathbf{x}_j^{ext}\end{bmatrix}=\underbrace{\begin{bmatrix}\mathbf{g}^{int}(\mathbf{V}^{int},\mathbf{V}^{ext})\\ \mathbf{g}^{ext}(\mathbf{V}^{int},\mathbf{V}^{ext},\mathbf{x}^{ext})\end{bmatrix}}_{\mathbf{g}} \tag{5}$$

where $\mathbf{A}_{1i}$ (resp. $\mathbf{D}_1$) represents the coupling between interior variables. $\mathbf{A}_{4i}$ (resp. $\mathbf{D}_4$) represents the coupling between local interface variables. $\mathbf{A}_{2i}$ and $\mathbf{A}_{3i}$ (resp. $\mathbf{D}_2$ and $\mathbf{D}_3$) represent the coupling between the local interface and the interior variables and, $\mathbf{B}_i$ (resp. $\mathbf{C}_j$) represent the coupling between the local interface variables and the external interface variables of the adjacent sub-domains.

### *3.2 Global Reduced System Formulation*

To formulate the global reduced system involving only the interface variables, the interior variables of the injector sub-domains are eliminated from (4), which yields for the *i*-th injector:

$$\mathbf{S}_i\triangle\mathbf{x}_i^{ext}+\mathbf{B}_i\triangle\mathbf{V}^{ext}=\tilde{\mathbf{f}}_i \tag{6}$$

where $\mathbf{S}_i=\mathbf{A}_{4i}-\mathbf{A}_{3i}\mathbf{A}_{1i}^{-1}\mathbf{A}_{2i}$ is the *local* Schur complement matrix and $\tilde{\mathbf{f}}_i=\mathbf{f}_i^{ext}-\mathbf{A}_{3i}\mathbf{A}_{1i}^{-1}\mathbf{f}_i^{int}$ the corresponding adjusted mismatch values.

Contrary to matrices $\mathbf{A}_i$, which are small but dense and general, matrix $\mathbf{D}$ is large but sparse and structurally symmetric. Thus, eliminating the interior variables from (5) would destroy its sparsity and symmetry. Therefore, all the variables of the network sub-domain are included in the reduced system (7).

$$\begin{bmatrix}\mathbf{S}_1 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{B}_1\\ \mathbf{0} & \mathbf{S}_2 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{B}_2\\ \mathbf{0} & \mathbf{0} & \mathbf{S}_3 & \cdots & \mathbf{0} & \mathbf{B}_3\\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{D}_1 & \mathbf{D}_2\\ \mathbf{C}_1 & \mathbf{C}_2 & \mathbf{C}_3 & \cdots & \mathbf{D}_3 & \mathbf{D}_4\end{bmatrix}\begin{bmatrix}\triangle\mathbf{x}_1^{ext}\\ \triangle\mathbf{x}_2^{ext}\\ \triangle\mathbf{x}_3^{ext}\\ \vdots\\ \triangle\mathbf{V}^{int}\\ \triangle\mathbf{V}^{ext}\end{bmatrix}=\begin{bmatrix}\tilde{\mathbf{f}}_1\\ \tilde{\mathbf{f}}_2\\ \tilde{\mathbf{f}}_3\\ \vdots\\ \mathbf{g}^{int}\\ \mathbf{g}^{ext}\end{bmatrix} \tag{7}$$

Due to the star layout of the decomposed system (see Fig. 1), the resulting global Schur complement matrix in (7) is block bordered diagonal. Manipulating this structure we can further eliminate all the interface variables of the injector sub-domains and keep only the variables associated to the network sub-domain, as shown in (8).

The elimination factors $\mathbf{C}_i\mathbf{S}_i^{-1}\mathbf{B}_i$ affect only non-zero elements of sub-matrix $\mathbf{D}_4$ thus retaining the original sparsity pattern. This system is solved efficiently using a sparse linear solver to update $\mathbf{V}$ at each Newton iteration. Then, the network interface variables ($\mathbf{V}^{ext}$) are backward substituted and the injector sub-domain variables ($\mathbf{x}_i$) are updated independently and in parallel using (4).

$$\underbrace{\begin{bmatrix} \mathbf{D}_1 & \mathbf{D}_2 \\ \mathbf{D}_3 & \mathbf{D}_4 - \sum_{i=1}^{N-1} \mathbf{C}_i \mathbf{S}_i^{-1} \mathbf{B}_i \end{bmatrix}}_{\widetilde{\mathbf{D}}} \underbrace{\begin{bmatrix} \Delta \mathbf{V}^{int} \\ \Delta \mathbf{V}^{ext} \end{bmatrix}}_{\Delta \mathbf{V}} = \underbrace{\begin{bmatrix} \mathbf{g}^{int} \\ \mathbf{g}^{ext} - \sum_{i=1}^{M-1} \mathbf{C}_i \mathbf{S}_i^{-1} \tilde{\mathbf{f}}_i \end{bmatrix}}_{\widetilde{\mathbf{g}}} \tag{8}$$

### *3.3 Exploiting Locality*

The procedure can be further accelerated by exploiting the locality of the sub-domains. Some sub-domains, described by strongly non-linear systems or with fast changing variables, converge slower. Other sub-domains, with "low dynamic activity", converge faster. This can be exploited in two ways.

First, subdomains with low dynamic activity are detected by measuring the effort (number of Newton iterations) needed for convergence at each discrete time. A subdomain's system is updated if that effort increases above a threshold. Second, a subdomain is declared converged (and stops being solved within the discrete time) if the absolute maximum normalized correction of a Newton solution of the subdomain system becomes smaller than a selected tolerance. Since the low dynamics are detected numerically during the simulation and the tolerance is chosen small enough so as not to disturb the Newton solution, the accuracy of the solution is preserved. Figure 2 shows the full parallel algorithm.

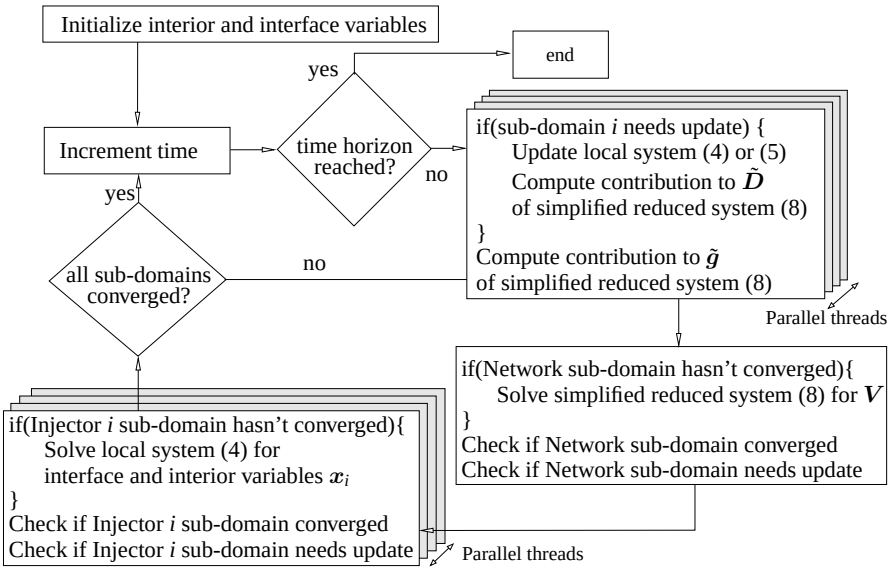



**Fig. 2** Parallel Algorithm (P)

## 4 Further Analysis of the Algorithm

To better understand its properties, Algorithm (P) in Fig. 2 can reformulated into an equivalent quasi-Newton undecomposed scheme with the $k$-th iteration described:

$$\underbrace{\begin{bmatrix} \mathbf{A}_1^{k_1} & \mathbf{0} & \cdots & \mathbf{0} & \widetilde{\mathbf{B}}_1^{k_1} \\ \mathbf{0} & \mathbf{A}_2^{k_2} & \cdots & \mathbf{0} & \widetilde{\mathbf{B}}_2^{k_2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A}_{M-1}^{k_{M-1}} & \widetilde{\mathbf{B}}_{M-1}^{k_{M-1}} \\ \widetilde{\mathbf{C}}_1^{k_M} & \widetilde{\mathbf{C}}_2^{k_M} & \cdots & \widetilde{\mathbf{C}}_{M-1}^{k_M} & \mathbf{D}^{k_M} \end{bmatrix}^k}_{\tilde{\mathbf{J}}^k} \underbrace{\begin{bmatrix} \triangle\mathbf{x}_1 \\ \triangle\mathbf{x}_2 \\ \vdots \\ \triangle\mathbf{x}_{M-1} \\ \triangle\mathbf{V} \end{bmatrix}^k}_{\triangle\mathbf{y}^k} = - \underbrace{\begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_{M-1} \\ \mathbf{g} \end{bmatrix}^k}_{\mathbf{F}^k} + \underbrace{\begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \vdots \\ \mathbf{r}_{M-1} \\ \mathbf{r}_\mathbf{M} \end{bmatrix}^k}_{\mathbf{r}^k}$$

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \triangle\mathbf{y}^k$$

where $0 \leq k_j \leq k$ $(j = 1, ..., M)$ and $\mathbf{r}_i = \begin{cases} \mathbf{f}_i, & \text{if } i\text{-th sub-domain has converged} \\ \mathbf{0}, & \text{otherwise.} \end{cases}$

The approximate Jacobian $\tilde{\mathbf{J}}^k$ is used by the method at each iteration $k$. Every block line $i$ of $\tilde{\mathbf{J}}^k$ corresponds to a sub-domain and is updated independently based on sub-domain update criteria [4]. Thus, some block lines can be kept constant for several iterations or even time-steps ($k_i \leq k$).

Furthermore, sub-domains considered to have converged are not solved any more (see Fig. 2). In the equivalent quasi-Newton integrated scheme this corresponds to explicitly setting the mismatch of those sub-domains to zero by introducing some inaccuracy to the method through the correction term $\mathbf{r}^k$. The inaccuracy is bounded and controlled to avoid affecting the accuracy of the final solution.

Using this formulation for Algorithm (P) allows us to utilize a general and well developed framework within which quasi-Newton schemes involving inaccuracy can be described and analyzed [12, 3].

## 5 Implementation and Numerical Results

The Schur Complement-based DDM was implemented in the simulation software RAMSES, developed at the University of Liège. The benchmark Algorithm (I) is a quasi-Newton scheme applied to the undecomposed DAE system (1). It uses an approximate Jacobian which is updated and factorized if the system hasn't converged after three Newton iterations at any discrete time instant. This method (also referred to as Very Dischonest Newton Method) is considered to be one of the fastest *sequential* algorithms and many traditional industry software use it.

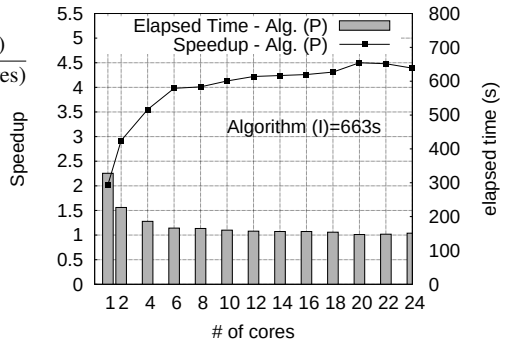
A large-scale model, representative of the Western European main transmission grid, is used. It includes 15226 buses, 21765 branches and 3483 synchronous machines represented in detail together with their excitation systems, voltage regu-

lators, power system stabilizers, speed governors and turbines. Additionally, 7211 models are included involving induction motors, dynamically modeled loads and equivalents of distribution systems. The resulting, undecomposed, DAE system has 146239 states. The disturbance simulated consists of a short circuit near a bus lasting 5 cycles (100 ms at 50 Hz), that is cleared by opening a double-circuit line. The system is then simulated over a period of 240 s with a time step of 1 cycle (20 ms).

**Fig. 3 Speedup index:**

$$\frac{\text{time elapsed sequential algorithm (I)}}{\text{time elapsed parallel algorithm (M cores)}}$$

This index shows how faster is the parallel implementation when compared to the fast sequential integrated Algorithm (I) on the same computer.




The same models, algebraization method (second-order Backward Differentiation Formula) and way of handling the discrete events are used in both algorithms. For the solution of the sparse linear systems, HSL MA41 [6] is used and for the dense injector linear systems of Algorithm (P), Intel MKL LAPACK library. The computer used for the simulation is a 24-core, shared memory, AMD Opteron Interlagos (CPU 6238 @ 2.60GHz) running Debian Linux.

**Fig. 4 Real-time index:**

$$n = \frac{\text{simulation elapsed time}}{\text{simulated physical time}}$$

This index shows how faster was the simulation than the simulated time. This is an important index for control center applications where the speed of computation is an issue for operator decision.

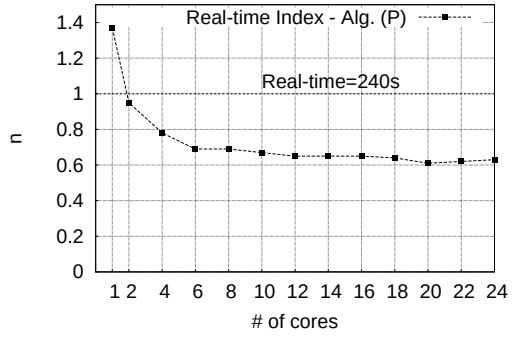



Figure 3 shows that the DDM-based algorithm is already twice faster than the benchmark in sequential execution. This speedup is mainly attributed to the exploitation of locality in the decomposed algorithm (Section 3.3). As we proceed to parallel execution, the proposed algorithm performs up to 4.5 times faster. Figure 4 shows the real-time potential of the algorithm in parallel execution.

The ratio of the interface system to the subdomain systems is very important to the performance of the algorithm since it corresponds to the ratio between the se-

quential portion of the code and the parallel portion of the code. A higher ratio leads to better speedup and avoids the saturation observed when increasing the number of cores. The size of the interface system (8) is the same as of the network subdomain (5), that is approx. 30 000 for the test-system considered. At the same time, the subdomain systems include approx. 120 000 states. Thus, the size ratio is approx. 4, which explains why a relatively small speedup is observed after 6 cores and the speedup saturates at 4.5 times.

## 6 Conclusion

In this paper a Schur Complement-based algorithm for dynamic simulation of electric power systems has been outlined. The algorithm yields acceleration of the simulation procedure in two ways. On the one hand, the procedure is accelerated numerically, by exploiting the locality of the sub-domain systems and avoiding many unnecessary computations (factorizations, evaluations, solutions). On the other hand, the procedure is accelerated computationally, by exploiting the parallelization opportunities inherent to DDMs.

## References

1. Crow, M., Ilic, M., White, J.: Convergence properties of the waveform relaxation method as applied to electric power systems. In: Circuits and Systems, 1989., IEEE International Symposium on, pp. 1863 –1866 vol.3 (1989)
2. CRSA, RTE, TE, TU/e: D4.1: Algorithmic requirements for simulation of large network extreme scenarios. Tech. rep. URL `http://www.fp7-pegase.eu/`
3. Dennis, J., Walker, H.: Inaccuracy in quasi-Newton methods: Local improvement theorems. Mathematical Programming at Oberwolfach II **22**, 70–85 (1984)
4. Fabozzi, D.: Decomposition, Localization and Time-Averaging Approaches in Large-Scale Power System Dynamic Simulation. Ph.D. thesis, University of Liège (2012)
5. Guibert, D., Tromeur-Dervout, D.: A Schur Complement Method for DAE/ODE Systems in Multi-Domain Mechanical Design. Domain Decomposition Methods in Science and Engineering XVII pp. 535–541 (2008)
6. HSL(2011): A collection of Fortran codes for large scale scientific computation. URL `http://www.hsl.rl.ac.uk`
7. Ilic'-Spong, M., Crow, M.L., Pai, M.A.: Transient Stability Simulation by Waveform Relaxation Methods. Power Systems, IEEE Transactions on **2**(4), 943–949 (1987)
8. Jackiewicz, Z., Kwapisz, M.: Convergence of waveform relaxation methods for differential-algebraic systems. SIAM Journal on Numerical Analysis **33**(6), 2303–2317 (1996)
9. Kron, G.: Diakoptics: the piecewise solution of large-scale systems. MacDonald (1963)
10. Kundur, P.: Power system stability and control. McGraw-hill New York (1994)
11. La Scala, M., Bose, A., Tylavsky, D., Chai, J.: A highly parallel method for transient stability analysis. Power Systems, IEEE Transactions on **5**(4), 1439 –1446 (1990)
12. Ortega, J., Rheinboldt, W.: Iterative Solution of Nonlinear Equations in Several Variables. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics (1987)
13. Saad, Y.: Iterative methods for sparse linear systems, second edn. Society for Industrial and Applied Mathematics (2003)