A Memory-Efficient Parallelizable Method for Computation of Thévenin Equivalents used in Real-Time Stability Assessment

Jørgensen, Christina Hildebrandt Lüthje; Møller, Jakob Glarbo; Sommer, Stefan; Jóhannsson, Hjörtur

Published in:
IEEE Transactions on Power Systems

Link to article, DOI:
10.1109/TPWRS.2019.2900560

Publication date:
2019

Document Version
Peer reviewed version

A Memory-Efficient Parallelizable Method for Computation of Thévenin Equivalents used in Real-Time Stability Assessment

Christina Hildebrandt Lüthje Jørgensen, *Student Member, IEEE,* Jakob Glarbo Møller, *Member, IEEE,* Stefan Sommer and Hjörtur Jóhannsson, *Member, IEEE*

***Abstract*—This paper introduces a factor-solve method, which efficiently computes Thévenin equivalents for all buses in the power system. A range of real-time stability assessment methods rely on Thévenin equivalents, and it is therefore essential that these can be determined fast and efficiently. The factor-solve method has runtime for computing Thévenin voltage that scales linearly with system size resulting in runtime of only a few milliseconds even for systems with several thousand buses. The computations only need the sparse admittance matrix for the power system and a sparse factorization resulting in low memory requirements, and furthermore Thévenin impedances can be determined in parallel. The factor-solve method is compared to a reference method, which uses coefficients for super-position to determine the Thévenin equivalents. The reference method is shown to have dissatisfying runtime and complexity. The factor-solve method is tested, parallelized and analysed, which shows a considerable speed-up in computations of Thévenin equivalents enabling them to be computed in real-time.**

***Index Terms*—Algorithms, Power system analysis computing, Real-time assessment, Thévenin equivalent**

NOMENCLATURE

X	Vector of complex entries
$\mathbf{X}$	Matrix with complex entries
$\lvert\mathbf{X}\rvert$	Number of non-zeros in matrix $\mathbf{X}$
$\mathcal{I}$	Identity matrix
$\mathcal{D}(X)$	Diagonalization of the vector X into a diagonal matrix

I. INTRODUCTION

THÉVENIN equivalent computations are used in assessment of power system stability. Steady-state stability of a generator can be determined as a margin of the maximum power injection using Thévenin equivalents [1], [2]. In [3]–[5] methods for voltage stability assessment are introduced which takes advantage of Thévenin equivalents, while [6] derives sensitivities based on a Thévenin equivalent representation to detect transient voltage sags. In [7] the Thévenin equivalent static contingency assessment (TESCA) method is introduced, which uses Thévenin equivalents to solve the power-flow problem and to evaluate aperiodic small signal stability following a contingency.

C. Jørgensen, J. Møller and H. Jóhannsson are with Center for Electric Power and Energy, Department of Electrical Engineering, Technical University of Denmark, Kongens Lyngby, Denmark (e-mail: chhil@elektro.dtu.dk).

S. Sommer is with Department of Computer Science, University of Copenhagen, Copenhagen, Denmark.

The introduction of methods for stability and security assessment as well as methods for defining countermeasures will either result in a competition for the available computational resources or introduce a need for larger computational resources. It is therefore essential, that the methods are first of all fast, but they should also use few computational resources (CPU, memory, etc.). Therefore, Thévenin equivalents should be computed both fast and efficiently. Decomposing the system using a Schur complement has previously been proposed in order to reduce system size and optimize computations.

In [8] the super-postion principle is used to determine the contribution of each voltage or current source to the Thévenin voltage. The method take advantage of a Schur complement, which [9] use to decompose the system for dynamic power system computations. In [5] a Schur complement is used to limit the computational burden when finding Thévenin equivalents for load buses, while [10] and the extended version of this [11] applies a Schur complement to efficiently compute Thévenin impedances for generators.

A range of methods for computing Thévenin equivalents for loads is analysed in [12] in order to assess a voltage stability margin. One method uses a Schur complement, and the complexity is estimated to have a cubic overhead, which is concluded to potentially become a burden for large systems. The Schur complement is generally considered to be dense [10], [13] and therefore it is computationally inefficient to determine. In reality all the coefficients for super-position and not just the Schur complement can be dense [14], which means that computing the coefficients is computational expensive.

In [14] different factorization methods are analysed to optimize the method for finding Thévenin equivalents. It is identified, that the Clark Kent LU factorization (KLU) factorization is the most efficient factorization method, however as in [8] it is determined that the greatest share of the execution time is spend on matrix multiplications due to the density of the matrices involved. The coefficients for super-position and Thévenin impedances are computed, whenever the topology of the power system change, whereas the Thévenin voltage can be determined for every new system state to monitor the system or for every iteration of a steady-state analysis. In [8] the method for determining coefficients for super-position was optimized, however the computation of Thévenin voltages was not addressed.

KLU is a factorization method optimized for sparse systems [15]. The method transform the system to block triangular

form and use approximate minimum degree ordering of the blocks to minimize fill-in before factorizing each block separately. KLU uses block back substitution to solve a system of linear equations. The factorization and the fill-in generated for sparse systems is close to linear with systems size in the context of this paper [10]. KLU is part of the library SuiteSparse [16].

This paper introduces a new method for computing Thévenin equivalents, opposed to the reference method [8], [14]. The method builds on the ideas given in [10] of developing a *factor-solve* method, where the method here will compute the Thévenin equivalents for the entire system instead of only the Thévenin impedances for generators. The method will take advantage of block back substitution in KLU to avoid computing coefficients for super-positon. Thereby, the computationally expensive matrix multiplication of the reference method will be omitted, and additionally the method will be considerably more memory-efficient.

Although [12] estimates a cubic runtime, when using a Schur complement, it will be shown that this is not the case for the methods in this paper. The *factor-solve* method will facilitate a speed-up of the runtime of both the computation of the Thévenin impedances as well as the Thévenin voltages. The method is split in to a sequential and parallel part compared to the sequential reference method, making it suitable for parallelization, which often enables the performance to become considerably better.

Following the introduction Section I.A gives some examples of real-time stability assessment methods using Thévenin equivalents. Section II describes the reference method for computing Thévenin equivalents using a Schur complement and investigate the runtime of this method. A *factor-solve* method for computing Thévenin equivalents is proposed in Section III. The method is implemented and tested in Section IV and lastly parallelized for optimal performance. The scalability of the method is evaluated as well as the resulting runtime and memory requirements of the computations. Section V discusses the results and gives perspectives on the method, while Section VI concludes the paper.

A. Thévenin equivalents in stability and security assessment

A range of methods for stability and security assessment use Thévenin equivalents in their computations. To ensure that these method can run in real-time, the Thévenin equivalent computations should be fast and efficient. Below follows some examples of Thévenin equivalents used in assessment methods.

1) Aperiodic small-signal rotor angle stability: In [1] the maximal injectable power by a generator is determined. This is used to determine a margin to the boundary for aperiodic small-signal rotor angle stability as a percentage margin to the maximal injectable power. This is based on algebraically derived equations [17] and given as

$$\%\Delta P_{inj} = \frac{P_{inj,max} - P_{inj}}{P_{inj,max}} \cdot 100\% \tag{1}$$

$$= \frac{\cos\left(\delta + \phi_{th}\right) + 1}{1 + \frac{|V|}{|V_{th}|}\cos\phi_{th}} \cdot 100\%, \tag{2}$$

where the generator is represented as a voltage source V with angle δ and the remaining grid by its Thévenin equivalent with a voltage source V_{th} and an impedance Z_{th} with angle ϕ_{th}. V_{th} is used as the phase angle reference.

A real-time remedial action can be computed as a countermeasure to keep the system from becoming unstable [2]. This is done by computing the reduction in power needed for the critical generator to become N-1 secure. The power is then redispatched to the remaining generators in the system ensuring these also remain secure.

2) Voltage stability: In [5] a voltage stability margin for the loads is determined. This uses the impedance match criterion to determine the maximum deliverable power to the load, which is given by

$$S_{max} = \frac{|V_{th}^2|\left[|Z_{th}| - \left(\text{imag}\left(Z_{th}\right)\sin\theta + \text{real}\left(Z_{th}\right)\cos\theta\right)\right]}{2\left[\text{imag}\left(Z_{th}\right)\cos\theta - \text{real}\left(Z_{th}\right)\sin\theta\right]^2} \tag{3}$$

with θ being the power factor angle of the load. The margin is then determined using the apparent power of the load S_i

$$\%\Delta S_i = \frac{S_{max,i} - S_i}{S_{max,i}} \cdot 100\%, \tag{4}$$

which determines the distance to the boundary of voltage stability.

3) Post-contingency aperiodic small-signal stability: In [7] a method for security assessment is described that determines the aperiodic small-signal stability of the power system following a contingency. The post-contingency steady-state nodal voltages is determined by first computing the Thévenin impedances post-contingency and then computing the Thévenin voltage and nodal voltage angle iteratively until the steady state voltage is found. The voltage angle δ_i is determined at each iteration as

$$\delta_i = \arccos\left(\frac{P_i|Z_{th,i}|}{|V_i||V_{th,i}|} - \frac{R_{th,i}|V_i|}{|Z_{th,i}||V_{th,i}|}\right) \tag{5}$$

The resulting steady-state nodal voltage can then be used to determine the N-1 stability using the earlier mentioned margin for maximum injectable power by generators.

II. Reference method

A. Schur complement and Thévenin equivalents

In [8], [14] a method for computing Thévenin equivalents is described, which uses coefficients for super-position. Thévenin equivalents consist of a Thévenin impedance Z_{th} and Thévenin voltage V_{th}. The Thévenin equivalent seen from node i satisfies

$$V_{th,i} = V_i - Z_{th,i}I_i, \tag{6}$$

where V_i is the node voltage and I_i is the current injected in to the network at node i.

Sources in a circuit can as in power-flow calculations be partitioned in to two sets - current sources (cs) and voltage sources (vs). Floating nodes may be represented as a current source injecting 0 current; loads may be represented as impedances, dependent- or independent current sources; generators with automatic voltage regulator (AVR) may be

represented as voltage sources; internal voltages of manually excited machines may be represented as voltage sources.

The admittance matrix for the system can then be block-wise partitioned as follows

$$\begin{bmatrix} I_{cs} \\ I_{vs} \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_{cs} & \mathbf{Y}_{v\to c} \\ \mathbf{Y}_{c\to v} & \mathbf{Y}_{vs} \end{bmatrix} \begin{bmatrix} V_{cs} \\ V_{vs} \end{bmatrix} \tag{7}$$

Eliminating V_{cs} from (7) yields

$$I_{vs} = \mathbf{Y}_{eq} V_{vs} - \mathbf{Q}_{ac} I_{cs} \tag{8}$$

with

$$\mathbf{Y}_{eq} = \mathbf{Y}_{vs} - \mathbf{Y}_{c\to v}\mathbf{Y}_{cs}^{-1}\mathbf{Y}_{v\to c} \tag{9}$$

$$\mathbf{Q}_{ac} = -\mathbf{Y}_{c\to v}\mathbf{Y}_{cs}^{-1} \tag{10}$$

where $\mathbf{Y}_{eq}$ is the Schur complement and $\mathbf{Q}_{ac}$ is the accompanying matrix. This reduction of the network is also known as *Kron reduction* [18].

The Thévenin impedances seen from node i is determined by short circuiting all voltage sources and open-circuiting all current sources, which will be

$$Z_{th,i} = \begin{cases} \mathbf{Z}_{cs}(i,i) & i \in cs \\ \mathbf{Y}_{eq}(i,i)^{-1} & i \in vs \end{cases} \tag{11}$$

where $\mathbf{Z}_{cs} = \mathbf{Y}_{cs}^{-1}$ [8].

Using the definition for Thévenin voltage given in (6) and the above network equations (8)-(10) the Thévenin voltages for the cs and vs nodes respectively are defined as

$$\begin{bmatrix} V_{th,cs} \\ V_{th,vs} \end{bmatrix} = \begin{bmatrix} \mathbf{Z}_c & \mathbf{K}_{v\to c} \\ \mathbf{Z}_{c\to v} & \mathbf{K}_v \end{bmatrix} \begin{bmatrix} I_{cs} \\ V_{vs} \end{bmatrix} = \mathbf{K} \begin{bmatrix} I_{cs} \\ V_{vs} \end{bmatrix} \tag{12}$$

with

$$\mathbf{Z}_c = \mathbf{Z}_{cs} - \mathcal{D}(Z_{th,cs}) \tag{13}$$

$$\mathbf{K}_{v\to c} = -\mathbf{Z}_{cs}\mathbf{Y}_{v\to c} \tag{14}$$

$$\mathbf{Z}_{c\to v} = \mathcal{D}(Z_{th,vs})\mathbf{Q}_{ac} \tag{15}$$

$$\mathbf{K}_v = \mathcal{I} - \mathcal{D}(Z_{th,vs})\mathbf{Y}_{eq} \tag{16}$$

Algorithm 1 determines the Thévenin impedances and the coefficient matrix $\mathbf{K}$ needed for computing the Thévenin voltages.

Algorithm 1 Thévenin equivalents

$\mathbf{L}_{cs}, \mathbf{U}_{cs} \leftarrow$ factorization of $\mathbf{Y}_{cs}$
$\mathbf{U}_{\mathbf{Z}_{cs}} \leftarrow$ solve($\mathbf{L}_{cs}, \mathcal{I}$)
$\mathbf{L}^T_{\mathbf{Z}_{cs}} \leftarrow$ solve($\mathbf{U}^T_{cs}, \mathcal{I}$)
$\mathbf{Z}_{cs} \leftarrow \mathbf{L}_{\mathbf{Z}_{cs}}\mathbf{U}_{\mathbf{Z}_{cs}}$
$Z_{th,cs} \leftarrow \mathcal{D}(\mathbf{Z}_{cs})$
$\mathbf{Q}_{ac} \leftarrow -\mathbf{Y}_{c\to v}\mathbf{Z}_{cs}$
$\mathbf{Y}_{eq} \leftarrow \mathbf{Y}_{vs} + \mathbf{Q}_{ac}\mathbf{Y}_{v\to c}$
$Z_{th,vs} \leftarrow \mathcal{D}(\mathbf{Y}_{eq})^{-1}$
$\mathbf{Z}_c \leftarrow \mathbf{Z}_{cs} - \mathcal{D}(Z_{th,cs})$
$\mathbf{K}_{v\to c} \leftarrow -\mathbf{Z}_{cs}\mathbf{Y}_{v\to c}$
$\mathbf{Z}_{c\to v} \leftarrow \mathcal{D}(Z_{th,vs})\mathbf{Q}_{ac}$
$\mathbf{K}_v \leftarrow \mathcal{I} - \mathcal{D}(Z_{th,vs})\mathbf{Y}_{eq}$
$Z_{th} \leftarrow \begin{bmatrix} Z_{th,cs} \\ Z_{th,vs} \end{bmatrix}$
$\mathbf{K} \leftarrow \begin{bmatrix} \mathbf{Z}_c & \mathbf{K}_{v\to c} \\ \mathbf{Z}_{c\to v} & \mathbf{K}_v \end{bmatrix}$
return Z_{th} and $\mathbf{K}$

B. Complexity of reference method

All computations of the reference method are done with sparse matrices and the complexity of the computations will therefore depend on the density of these matrices.

Computing $\mathbf{Z}_{cs}$ will have the complexity $O(|\mathbf{L}_{\mathbf{Z}_{cs}}||cs|)$, which is the most expensive computation in Algorithm 1. By comparison inverting sparse matrices will at maximum have one computation per non-zero element in the matrix for each column in the identity matrix, which gives a complexity of $O(|\mathbf{L}_{cs}||cs|) = O(|\mathbf{U}_{cs}||cs|)$ for solve($\mathbf{L}_{cs}, \mathcal{I}$) and solve($\mathbf{U}^T_{cs}, \mathcal{I}$). Empirically $|\mathbf{U}_{cs}| \leq |\mathbf{L}_{\mathbf{Z}_{cs}}|$, and therefore inversion will be computationally cheaper than computing $\mathbf{Z}_{cs}$. The remaining computations of Algorithm 1 will also be of lower complexity, since they involve at least one sparse matrix, and as stated earlier the factorization is close to linear in complexity. Hence the complexity of the algorithm will be $O(|\mathbf{L}_{\mathbf{Z}_{cs}}||cs|)$.

Assuming that $|\mathbf{L}_{\mathbf{Z}_{cs}}|$ ($\simeq |\mathbf{U}_{\mathbf{Z}_{cs}}|$) scale close to linear with system size like it's the case in practise with the number of non-zeros and fill-in generated in the factorization in this context [10] the complexity will be $O(|cs|^2)$.

Determining Thévenin voltages (12) is $O(|\mathbf{K}|)$, and therefore the computation of Thévenin voltages will depend on the density of the coefficient matrix.

C. Test of reference method

The reference method for determining Thévenin equivalents using Algorithm 1 and equation (12) is analysed in MATLAB on a CPU of Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz. In the implementation KLU is used as the factorization method, due to its efficiency for sparse systems.

The test systems are given in Table I. The Pegase and Polish-Winter systems can be found in MATPOWER [19], the PTI systems are included in the PSS®E 33.0 examples and Nordic32 can be found in [20]. EECC-PSSE-33-0 is a representation of the American Eastern interconnection.

TABLE I
TEST SYSTEMS

Case	no. of buses	no. of vs nodes	non-zeros in $\mathbf{Y}$
Nordic32	46	20	160
Pegase89	89	12	501
Pegase1354	1354	260	4774
PTI-WECC-1648	1648	313	6680
Polish-Winter99	2383	327	8155
Polish-Winter03	2746	374	9344
Pegase2869	2869	510	10805
Polish-Winter07	3012	347	10144
PTI-EECC-7991	7917	1325	32211
Pegase9241	9241	1445	37655
Pegase13659	13659	4092	50909
EECC-PSSE-33-0	29827	3780	107527

Table II shows the density of the coefficient matrix for each test system. Here the density is given as the number of non-zeros and as the percentage of non-zeros to the maximum size of the matrix. Table III shows the resulting runtime.

TABLE II
DENSITY OF $\mathbf{K}$ FOR TEST SYSTEMS

Case	non-zeros in $\mathbf{K}$	density of $\mathbf{K}$ (%)
Nordic32	547	25.9
Pegase89	7663	96.7
Pegase1354	1120078	61.1
PTI-WECC-1648	1706580	62.8
Polish-Winter99	2825717	49.8
Polish-Winter03	3027555	40.2
Pegase2869	2961793	36.0
Polish-Winter07	4206582	46.4
PTI-EECC-7991	44852962	71.6
Pegase9241	42515659	49.8
Pegase13659	184786127	99.0
EECC-PSSE-33-0	854782395	96.1

TABLE III
RUNTIME OF ALGORITHM 1 AND OF COMPUTING THÉVENIN VOLTAGES

Case	Runtime (s) Algorithm 1	Runtime (ms) V_{th}
Nordic32	$2.98{\cdot}10^{-4}$	$1.27{\cdot}10^{-2}$
Pegase89	$1.34{\cdot}10^{-3}$	$3.13{\cdot}10^{-2}$
Pegase1354	0.11	2.74
PTI-WECC-1648	0.20	4.62
Polish-Winter99	0.30	7.08
Polish-Winter03	0.31	8.50
Pegase2869	0.41	7.65
Polish-Winter07	0.42	10.51
PTI-EECC-7991	6.31	118.13
Pegase9241	11.58	118.95
Pegase13659	36.70	530.13
EECC-PSSE-33-0	253.08	2519.75

For small systems the method is viable and the algorithm has a runtime of a few milliseconds, however as the system size and complexity grows, so does the runtime. Figure 1 shows the runtime of Algorithm 1 and the runtime for computing Thévenin voltages V_{th} (12) plotted against system size.

As expected, the figure shows complexity that is close to quadratic to system size for Algorithm 1. The complexity is dependent on $|\mathbf{L}_{\mathbf{Z}_{cs}}|$, which was assumed to scale close to linear with $|cs|$. $|cs|$ scales close to linear with system size as seen in Table I. This results in an almost quadratic complexity.

The complexity of computing Thévenin voltages was analysed to be $O(|\mathbf{K}|)$. An increased system size result in larger matrices and thereby also a possibility of a larger number of non-zeros, therefore the close to quadractic tendency for these computations is reasonable.

PTI-EECC-7991 and Pegase9241 has almost the same number of non-zeros in the coefficient matrix $|\mathbf{K}|$ even though the system size for Pegase9241 is considerably larger. As expected this gives nearly identical runtimes for computing V_{th}. Furthermore, the number of non-zeroes $|\mathbf{K}|$ for EECC-PSSE-33-0 is almost 5 times larger than for Pegase13659 resulting in a runtime for V_{th}, which is also 5 times larger. This is consistent with the analysed complexity.

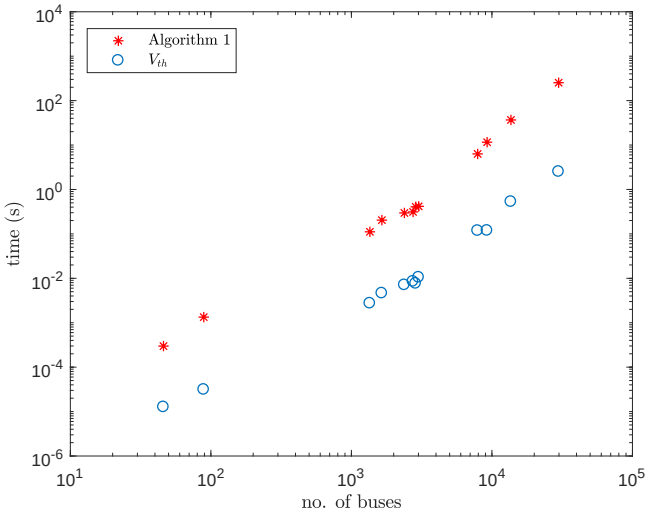

Fig. 1. Runtime for the initial method depending on the number of buses. The runtime is shown for Algorithm 1 and the Thévenin voltages (12) and the plot is logarithmic.

III. INTRODUCTION OF FACTOR-SOLVE METHOD

Performance of Algorithm 1 and the computation of Thévenin voltages is dissatisfying and doesn't scale well with system size, but it turns out that there is a potential for improvements. KLU solves a system by using block back substitution. By use of this solver it is possible to find the Thévenin voltages without computing the coefficient matrix.
The equations from (12) can be written as

$$V_{th,cs} = \left(\mathbf{Z}_{cs} - \mathcal{D}\left(Z_{th,cs}\right)\right) I_{cs} - \mathbf{Z}_{cs}\mathbf{Y}_{v\to c}V_{vs} \quad (17)$$

$$V_{th,vs} = \mathcal{D}(Z_{th,vs})\mathbf{Q}_{ac}I_{cs} + \left(\mathcal{I} - \mathcal{D}(Z_{th,vs})\mathbf{Y}_{eq}\right)V_{vs} \quad (18)$$

Defining $\tilde{V}$ as

$$\tilde{V} = \mathbf{Z}_{cs}\left(I_{cs} - \mathbf{Y}_{v\to c}V_{vs}\right) \quad (19)$$

and inserting this in to (17) and (18) gives

$$V_{th,cs} = \tilde{V} - Z_{th,cs}I_{cs} \quad (20)$$

$$V_{th,vs} = V_{vs} - Z_{th,vs}\left(\mathbf{Y}_{vs}V_{vs} + \mathbf{Y}_{c\to v}\tilde{V}\right) \quad (21)$$

Solving $\mathbf{Y}_{cs}x = b$ for x i.e. finding $x = \mathbf{Z}_{cs}b$ can be determined by block back substitution using the factors LU from the KLU factorization. This will be defined as klu(LU, b) and $\tilde{V}$ can then be calculated by klu$(LU, I_{cs} - \mathbf{Y}_{v\to c}V_{vs})$. It is hereby possible to determine the Thévenin voltages using only the factorization of $\mathbf{Y}_{cs}$ and the Thévenin impedances. This means, that the entire coefficient matrix $\mathbf{K}$ is no longer needed, which will simplify the algorithm.

The Thévenin impedances for cs and vs nodes are defined in (11). The diagonal of $\mathbf{Z}_{cs}$ and the diagonal of the Schur complement $\mathbf{Y}_{eq}$ is needed in these computations.

The diagonal of $\mathbf{Z}_{cs}$ determines the Thévenin impedances for cs nodes and is computed by taking $\mathbf{U}_{\mathbf{Z}_{cs}} = \mathbf{L}_{cs}^{-1}$ and $\mathbf{L}_{\mathbf{Z}_{cs}} = \mathbf{U}_{cs}^{-1}$ and multiplying the rows and columns, that result in the diagonal i.e.

$$Z_{th,cs,k} = \mathbf{L}_{\mathbf{Z}_{cs}}(k,:)\mathbf{U}_{\mathbf{Z}_{cs}}(:,k) \quad \forall k \in cs, \quad (22)$$

where $\mathbf{L}_{\mathbf{Z}_{cs}}(k,:)$ is the k'th row of $\mathbf{L}_{\mathbf{Z}_{cs}}$ and $\mathbf{U}_{\mathbf{Z}_{cs}}(:,k)$ is the k'th column of $\mathbf{U}_{\mathbf{Z}_{cs}}$.

The Thévenin impedances for the vs nodes are given as the inverse of the diagonal elements of the Schur complement $\mathbf{Y}_{eq}$.

$$Z_{th,vs,k} = \mathbf{Y}_{eq}(k,k)^{-1} \quad \forall k \in vs, \tag{23}$$

which is scalar inversion.

Using (9) this can be determined by

$$\mathbf{Y}_{eq}(k,k) = \mathbf{Y}_{vs}(k,k) - \mathbf{Y}_{c\to v}(k,:)\mathbf{Z}_{cs}\mathbf{Y}_{v\to c}(:,k) \tag{24}$$

As with the Thévenin voltages block back substitution is used to determine part of the equation. The Thévenin impedances for the vs nodes are found as

$$\hat{\mathbf{U}}(:,k) = \mathbf{Z}_{cs}\mathbf{Y}_{v\to c}(:,k) \leftarrow \text{klu}\left(LU, \mathbf{Y}_{v\to c}(:,k)\right) \tag{25}$$

$$\mathbf{Y}_{eq}(k,k) = \mathbf{Y}_{vs}(k,k) - \mathbf{Y}_{c\to v}(k,:)\hat{\mathbf{U}}(:,k) \tag{26}$$

$$Z_{th,vs,k} = \mathbf{Y}_{eq}(k,k)^{-1} \tag{27}$$

The computations for the vs nodes is similar to the computations given in [10], and the method will therefore be called a *factor-solve* method. The method factorize part of the system and then solves for part of the equations to efficiently compute the solution. This approach is used to find the Thévenin impedances for the vs nodes and to find the Thévenin voltages for the entire system.

Algorithm 2 factorize $\mathbf{Y}_{cs}$ and compute the Thévenin impedances possibly in parallel.

Algorithm 2 Thévenin equivalents

$\mathbf{L}_{cs}, \mathbf{U}_{cs} \leftarrow$ factorization of $\mathbf{Y}_{cs}$ {Output: LU}
$\mathbf{U}_{\mathbf{Z}_{cs}} \leftarrow$ solve($\mathbf{L}_{cs}, \mathcal{I}$)
$\mathbf{L}_{\mathbf{Z}_{cs}} \leftarrow$ solve($\mathbf{U}_{cs}^T, \mathcal{I}$)^T
for $k = 1..|cs|$ {In parallel} **do**
  $Z_{th,cs,k} \leftarrow \mathbf{L}_{\mathbf{Z}_{cs}}(k,:)\mathbf{U}_{\mathbf{Z}_{cs}}(:,k)$
end for
for $k = 1..|vs|$ {In parallel} **do**
  $\hat{\mathbf{U}}(:,k) \leftarrow$ klu($LU, \mathbf{Y}_{v\to c}(:,k)$)
  $\mathbf{Y}_{eq}(k,k) \leftarrow \mathbf{Y}_{vs}(k,k) - \mathbf{Y}_{c\to v}(k,:)\hat{\mathbf{U}}(:,k)$
  $Z_{th,vs,k} \leftarrow \mathbf{Y}_{eq}(k,k)^{-1}$
end for
$Z_{th} \leftarrow \begin{bmatrix} Z_{th,cs} \\ Z_{th,vs} \end{bmatrix}$
return Z_{th} and LU

This way no computation time is spent on the coefficients and furthermore the computation of the Thévenin voltages is altered from matrix vector multiplication with the dense coefficient matrix to a block back substitution and matrix vector multiplications with sparse matrices.

A. Complexity of factor-solve method

The complexity of Algorithm 2 is split in to a sequential and a parallel part. The first part is sequential and will be $O(|cs|^2)$, since the inversions are $O(|\mathbf{L}_{cs}||cs|)$ and $O(|\mathbf{U}_{cs}||cs|)$ respectively and the fill-in scales linearly with system size. The factorization has negligible runtime compared to this due to the close to linear complexity.

The two loops run sequential will be $O(|cs|^2)$ and $O(|cs||vs|)$, due to the linear complexity of the computations in the loops. When run in parallel the runtime is theoretically determined by the number of cores. By Amdahl's law the total runtime will only be limited by the sequential part of the algorithm, since an unlimited number of cores can be added to completely parallelize the loops. In practise however there will be an overhead due to communication between the cores.

For the computation of the Thévenin voltages the complexity will be $O(|cs|)$. The matrix-vector multiplication is linear due to the sparsity of the matrices scaling with system size, and the block back substitution of KLU is close to linear. This complexity is lower than the reference method's complexity of $O(|\mathbf{K}|)$, which is almost quadratic to system size for the largest systems.

IV. Implementation and test of factor-solve method

The *factor-solve* method is implemented in MATLAB to evaluate the method with respect to runtime of both Algorithm 2 and the computation of Thévenin voltages as well as the accuracy of the results and the memory requirements. The method is tested on Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz. The test systems were given earlier in Table I.

The resulting runtime for Algorithm 2 and the runtime for computing the Thévenin voltages (19)-(21) is shown in Table IV, while Fig. 2 shows a plot of the runtime.

TABLE IV
Runtime and speed-up and error for factor-solve method

Case	Runtime (s) Algorithm 2	Speed-up Algorithm 2	Runtime (ms) V_{th}	Speed-up V_{th}
Nordic32	0.045	0.01	0.084	0.15
Pegase89	0.050	0.03	0.101	0.31
Pegase1354	0.114	0.97	0.295	9.28
PTI-WECC-1648	0.544	0.37	0.731	6.32
Polish-Winter99	0.241	1.23	0.457	15.50
Polish-Winter03	0.319	0.98	0.540	15.72
Pegase2869	0.355	1.15	0.604	12.66
Polish-Winter07	0.300	1.39	0.565	18.60
PTI-EECC-7991	1.745	3.62	1.473	80.18
Pegase9241	2.983	3.88	1.708	69.63
Pegase13659	6.276	5.85	2.328	227.75
EECC-PSSE-33-0	20.592	12.29	5.522	456.28

Algorithm 2 is seen to have close to quadratic complexity as analysed earlier, while the Thévenin voltages is seen to have an almost linear complexity as analysed earlier. The change in complexity for the Thévenin voltages result in a significant decrease in runtime compared to the reference method. It is notable that, all test systems have runtimes for computing Thévenin voltages below 6 ms. The system PTI-WECC-1648 has a runtime that is considerably different compared to the other systems. The complexity is only close to linear and actually depends on the fill-in in the factorization, which can differ depending on the structure of the system.

The speed-up is calculated as a quantity for the performance of the *factor-solve* method compared to the reference method. This is defined as $\frac{t_1}{t_2}$, where t_1 is the runtime of the reference

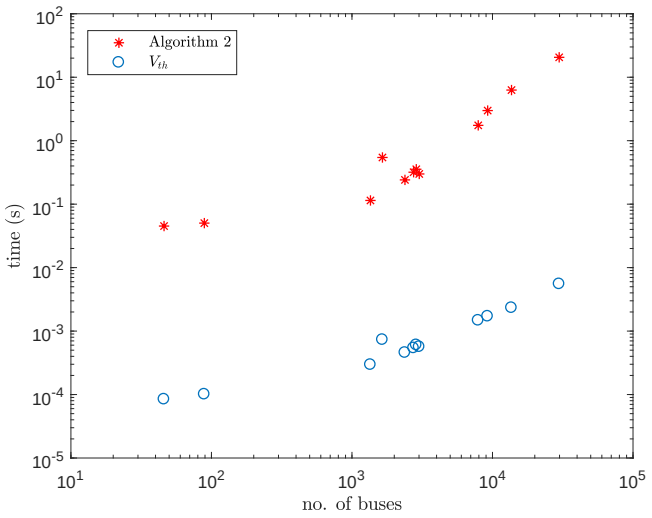

Fig. 2. Runtime for the *factor-solve* method depending on the number of buses. The runtime is shown for Algorithm 2 and the Thévenin voltages (12) and the plot is logarithmic.

method and t_2 is the runtime of the *factor-solve* method. The speed-up is shown alongside the runtimes in Table IV.

For the smaller systems neither the algorithm nor the calculations of Thévenin voltage receives a speed-up. Systems larger than 1000 buses is sped up in Thévenin voltage computations but only some benefit from Algorithm 2. Algorithm 2 requires large power systems to be faster than Algorithm 1.

Some systems benefit more from computing Thévenin voltages with the *factor-solve* method instead of the reference method. The systems Pegase2869 and Pegase9241 both have a lower speed-up than test systems of similar size. The reason for this is found in Table II. The coefficient matrix for both Pegase2869 and Pegase9241 is less dense than for the systems with similar size, and therefore weren't as slow with the reference method. The complexity has changed from being dependent on the number of non-zeros in the coefficients to be close to linearly dependent on system size, which now gives runtimes, that scale better with system size.

Errors in Thévenin voltages obtained with the two methods can be stated in terms of a total vector error (TVE) [21] using the reference method with the standard LU factorization in MATLAB (UMFPACK) as reference

$$\text{TVE } (\%) = \sqrt{\frac{(\tilde{X}_r - X_r)^2 + (\tilde{X}_i - X_i)^2}{X_r^2 + X_i^2}} \cdot 100\%, \quad (28)$$

where $\tilde{X}$ is the estimate (reference or *factor-solve* method) and X is the true value (reference method with UMFPACK).

The maximum TVE can be seen in Table V. The two methods only differs by a small margin in accuracy of the resulting Thévenin voltages.

A benefit from the *factor-solve* method is the amount of memory needed. The reference method need to store the Thévenin impedances along with the coefficient matrix $\mathbf{K}$, while the *factor-solve* method need to store the Thévenin impedances and the sparse factorization of $\mathbf{Y}_{cs}$. Especially for the larger systems there is a significant reduction in memory using the *factor-solve* method, since the coefficient matrix is

TABLE V
MAXIMUM TVE (%) FOR THE REFERENCE AND FACTOR-SOLVE METHOD

Case	Max TVE (%) (reference)	Max TVE (%) (*factor-solve*)
Nordic32	$2.43{\cdot}10^{-13}$	$2.23{\cdot}10^{-13}$
Pegase89	$1.71{\cdot}10^{-12}$	$1.71{\cdot}10^{-12}$
Pegase1354	$4.55{\cdot}10^{-12}$	$4.56{\cdot}10^{-12}$
PTI-WECC-1648	$6.98{\cdot}10^{-12}$	$7.02{\cdot}10^{-12}$
Polish-Winter99	$1.95{\cdot}10^{-11}$	$2.79{\cdot}10^{-11}$
Polish-Winter03	$2.98{\cdot}10^{-11}$	$2.98{\cdot}10^{-11}$
Pegase2869	$5.50{\cdot}10^{-12}$	$5.54{\cdot}10^{-12}$
Polish-Winter07	$1.52{\cdot}10^{-11}$	$1.52{\cdot}10^{-11}$
PTI-EECC-7991	$9.26{\cdot}10^{-12}$	$9.17{\cdot}10^{-12}$
Pegase9241	$2.13{\cdot}10^{-11}$	$2.13{\cdot}10^{-11}$
Pegase13659	$2.20{\cdot}10^{-11}$	$2.22{\cdot}10^{-11}$
EECC-PSSE-33-0	$1.30{\cdot}10^{-09}$	$8.01{\cdot}10^{-10}$

dense. Sparse matrices store the location and value of a non-zero entry, while full matrices store all entries of a matrix. The test systems with a coefficient matrix with a high density, see Table II, will therefore be more efficiently stored as a full matrix than a sparse.

The KLU factorization of a matrix $\mathbf{A}$ is defined as

$$\mathbf{PRAQ} = \mathbf{LU} + \mathbf{F} \quad (29)$$

$\mathbf{P}, \mathbf{Q}$ are permutations stored as vectors, $\mathbf{R}$ is a scaling matrix optimally stored as a vector, $\mathbf{L}, \mathbf{U}$ are complex sparse matrices and $\mathbf{F} = 0$ in this context [14].

The memory requirements for storing the the coefficient matrix (either in full or sparse format) compared to storing the sparse factorization can be seen in Table VI. Integers like doubles are stored using 64 bits.

TABLE VI
MEMORY REQUIREMENTS FOR COEFFICIENT MATRIX $\mathbf{K}$ IN SPARSE AND FULL FORMAT AND FOR THE FACTORIZATION OF $\mathbf{Y}_{cs}$

Case	$\mathbf{K}$ (full)	$\mathbf{K}$ (sparse)	Factorization of $\mathbf{Y}_{cs}$
Nordic32	33.1 kB	13.2 kB	2.9 kB
Pegase89	123.8 kB	180.3 kB	17.7 kB
Pegase1354	28.0 MB	25.6 MB	164.3 kB
PTI-WECC-1648	41.4 MB	39.1 MB	233.6 kB
Polish-Winter99	86.6 MB	64.7 MB	298.8 kB
Polish-Winter03	115.1 MB	69.3 MB	349.4 kB
Pegase2869	125.6 MB	67.8 MB	390.5 kB
Polish-Winter07	138.4 MB	96.3 MB	400.4 kB
PTI-EECC-7991	0.9 GB	1.0 GB	1.2 MB
Pegase9241	1.3 GB	1.0 GB	1.4 MB
Pegase13659	2.8 GB	4.1 GB	2.1 MB
EECC-PSSE-33-0	13.3 GB	19.1 GB	5.6 MB

Storing the factorization requires less memory than storing the coefficient matrix for all test systems i.e. the *factor-solve* method requires less than the reference method. As with the runtime for the Thévenin voltages it is also here the largest systems that have the biggest improvement. For the test system EECC-PSSE-33-0, which is optimally stored as a full matrix, the improvement is a factor of over 2400, while the factor is around 220 for Polish-Winter99 and only 4.6 for Nordic32.

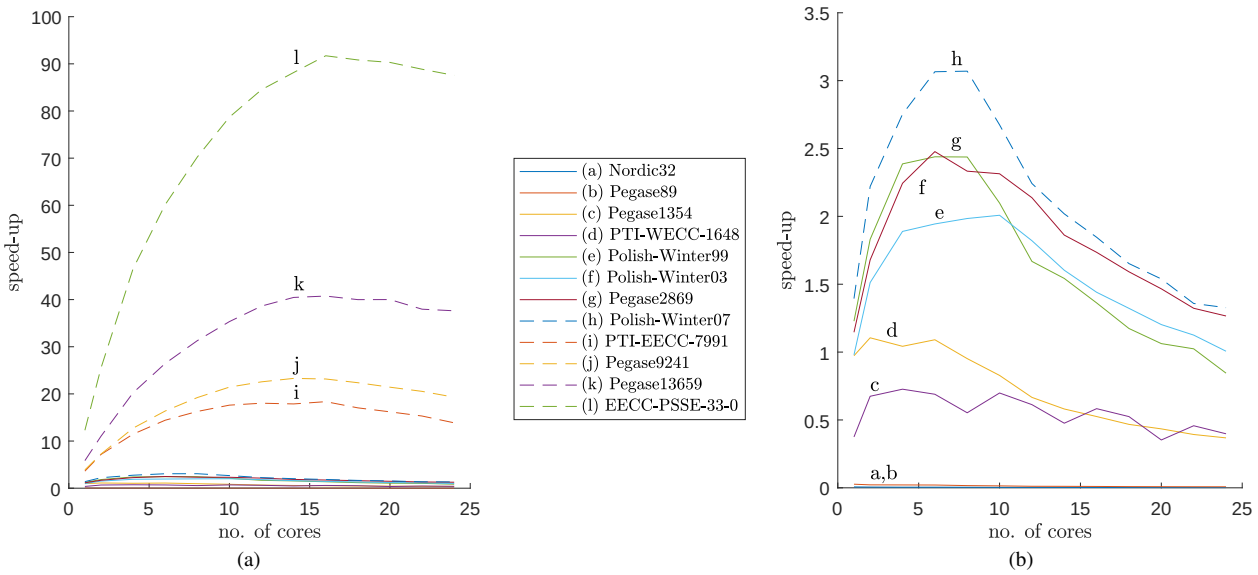

Fig. 3. Speed-up of Algorithm 2 compared to Algorithm 1 for all test systems - (a) shows all test systems and (b) is a zoom of the smaller systems.

This difference is due to the scaling of the memory. The memory for storing $\mathbf{K}$ as a sparse matrix is scaling close to quadratic with system size, while the memory for the sparse factorization of $\mathbf{Y}_{cs}$ is scaling linearly.

A. Parallelization of Algorithm 2

The runtime for computing Thévenin voltages has been decreased significantly, however Algorithm 2 is only considerable faster for the larger systems. A benefit from the *factor-solve* method is that Algorithm 2 can easily be parallelized. The runtime is therefore tested on a machine with 2 CPUs of Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz with 12 cores each. The algorithm will be tested on the following number of cores $1, 2, 4, 6, \ldots, 22, 24$.

Fig. 3a shows the speed-up of Algorithm 2 compared to Algorithm 1 for different number of cores for each test system and Fig. 3b shows the same with the 4 largest systems excluded.

Parallelization decrease the runtime of Algorithm 2 considerably. As system size grows the speed-up increases significantly up to a factor of 90 for the largest system compared to a factor 12 without parallelization. The smallest systems still has no speed-up, since these systems already have a short runtime due to their limited size, and the introduction of loops and overhead time only slows them down. There is a range of test systems that when run sequential did not benefit from Algorithm 2, however when parallelized there is now a benefit. Fig. 3a and 3b furthermore show, that the optimal number of cores increase with system size. After the point of maximal speed-up it decreases due to the increased overhead time, which will be larger than the gain of adding additional cores.

The for loops in Algorithm 2 are implemented with the function `parfor` in Matlab for all cores. This will for 1 core i.e. the sequential version be a little slower than using `for`, since there is a small penalty when using `parfor`. The sequential version could therefore be a little faster.

Table VII shows the number of cores that maximize the speed-up for Algorithm 2 and thereby also minimize the runtime. It shows explicitly that the optimal number of cores increase with systems size and so does the speed-up. Table VII show a runtime of 2.8s for EECC-PSSE-33-0 with the parallelized *factor-solve* method compared to 253s for the reference method. This means that the *factor-solve* method will be able to respond faster to a sudden change in the system topology than the reference method.

TABLE VII
RESULTS FOR THE OPTIMAL NUMBER OF CORES FOR EACH TEST SYSTEM

Case	Optimal no. of cores	Runtime (s) Algorithm 2	Speed-up Algorithm 2
Nordic32	1	0.045	0.007
Pegase89	1	0.050	0.027
Pegase1354	2	0.101	1.105
PTI-WECC-1648	4	0.281	0.727
Polish-Winter99	6	0.121	2.439
Polish-Winter03	10	0.155	2.008
Pegase2869	6	0.164	2.477
Polish-Winter07	8	0.136	3.070
PTI-EECC-7991	16	0.344	18.354
Pegase9241	14	0.497	23.289
Pegase13659	16	0.901	40.750
EECC-PSSE-33-0	16	2.759	91.715

Fig. 4 show the distribution of runtime on each part of Algorithm 2 for the optimal number of cores given in Table VII. Factorization time is negligible as expected by its almost linear complexity compared to the quadratic complexity of the entire algorithm as for the reference method [14].

The majority of time is spent on computing the Thévenin impedances for the cs nodes. However, there is also a significantly larger number of cs nodes than vs nodes as seen in Table I. Dividing the runtimes by the number of cs and vs nodes respectively, gives an average runtime for cs nodes that is lower than for the vs nodes in every case. The reason might be found in the difference in the computations. However, it might also be that the optimal number of cores for the

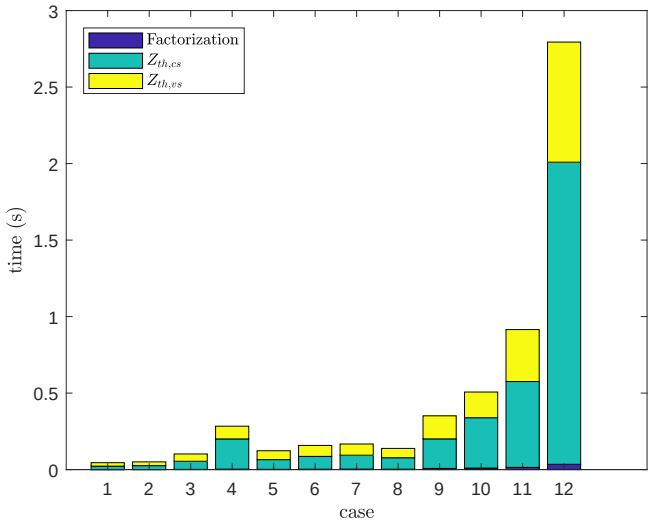

Fig. 4. Distribution of the runtime of Algorithm 2 on to each part of the algorithm for the optimal number of cores.

cs nodes is larger than for the vs nodes. The loops are independent, and therefore running both loops on the same number of cores might result in the computations for the vs nodes being more dominated by overhead. Computations for cs nodes are furthermore split in to a sequential part (inversion of factorization factors) and a parallel part, where the first part will not benefit from additional cores.

V. Discussion and Perspectives

The *factor-solve* method determines Thévenin voltages in linear time given the Thévenin impedances and the factorization of the admittance matrix for the cs nodes. This means that on the given CPU the Thévenin voltages for all buses in the system could be determined in under 6 ms for tests system up to a size of 30.000 buses. If the calculations where to be used in connection with Phasor Measurement Units (PMUs) [22], [23] the method would be able to determine the Thévenin voltages for every measurement, since these will normally be received at the rate of the system frequency (every 16-20ms).

In the *factor-solve* method the runtime of Algorithm 2 dominates the computations, but this is only run, whenever the system topology change. The factorization time is negligible due to linearity, thus the majority of time is spent on determining the Thévenin impedances. Runtime for the *factor-solve* method compared to the reference method is better especially for the larger systems. The complexity is still quadratic as with the reference method, however the implementation is a little faster. More importantly the method can easily be parallelized, since the *factor-solve* method is split in to a sequential and a parallel part compared to the reference method consisting of sequential matrix multiplications.

The fact that the algorithm can be parallelized enables the method to have an even lower runtime. Using only a couple of cores makes the method better than the initial method for systems with 2000 buses or more, while a single core is sufficient to decrease runtime for the larger systems. The system size and the density of the larger systems furthermore increases the gain from using parallelization.

The smallest systems does not benefit from the *factor-solve* method, since these are so small that the time spend on matrix multiplications is neglible and changing these computations in to loops only worsen the runtime due to the overhead.

A few systems benefit from computing Thévenin voltages by use of the *factor-solve* method without benefiting from Algorithm 2 even with parallelization. An alternative method for these system would be to combine the two methods. The Thévenin impedances along with the factorization could be computed by line 1-8 in Algorithm 1 and then the Thévenin voltages would be computed by the *factor-solve* method. These systems will then get the lowest possible runtime, and furthermore only running line 1-8 of Algorithm 1 will also result in a further decrease in the runtime of computing the Thévenin impedances.

This sort of hybrid method would be useable for systems consisting of between 1.000 and 2.000 busses. It should however be noted that the runtime for both methods for these systems is low enough for real-time computations and either one would be suitable. The method can be used for security assessment in for example the Thévenin equivalent static contingency assessment method [7]. Here Thévenin impedances is determined for all N-1 contingencies and Thévenin voltages are then computed several times when determining the steady-state nodal voltage. For contingency analysis it is important to use the fastest combination to ensure that assessment can be done in reasonable time.

An idea for future work would be to investigate the potential use of GPUs instead of doing all the computations on the CPU. Moving data to and from the GPU is expensive, but when the data is there the computations can be executed and parallelized more efficiently. It could potentially be used on the loops in Algorithm 2. However, since there is no reuse of data in the loops it might not give a better performance, since a GPU excel when doing the same computations multiple times. Furthermore, it would also be satisfactory if the sequential part of the algorithm could have it's complexity reduced in order to scale better or be changed to be able to run in parallel.

In the complexity analysis the runtime was determined to be dependent on the sparsity of the factors in the factorization, which is almost linear to the system size. From the results it can be seen that this is the case for most of the test systems, however the structure of some systems give rise to a larger fill-in and will affect the resulting runtime. It would be interesting to find specific reasons for the behaviour of these system.

A clear benefit from the factor-solve method is the decrease in memory usage, where a reduction in required memory was seen for all test systems. This together with the now linear complexity of the Thévenin voltages computations will make the method use fewer computational resources and thereby leave room for other assessment methods.

Future work would also be to include the *factor-solve* method in stability assessment methods to test their performance with the new calculations. It would then be possible to analyse the these methods and determine new areas suitable for optimization. It would furthermore be interesting to test the method in a real-time setting on a SW-platform like [24].

VI. Conclusion

The paper describes a reference method for determining the Thévenin equivalents for all nodes in a power system. The reference method was analysed to be insufficient especially for large power systems. The given complexity of the method is dependent on the density of the coefficients and will therefore be less viable for some systems compared to others due to the structure of the power system.

A *factor-solve* method was introduced, which takes advantage of the block back substitution in KLU. The method spends no computation time on generating coefficients and the calculations of Thévenin voltages computation can be done in linear time. Furthermore, the memory usage is significantly lower than for the reference method changing from gigabytes to a couple of megabytes for larger systems.

Computations of Thévenin impedances can take advantage of parallelization and runtime will therefore be dependent on the number of cores used. The optimal number of cores is shown to increase with system size. The runtime for determining Thévenin impedances is still not satisfying for the largest test system, however these will only be determined, when system topology change. Thévenin voltages are determined without relying on parallization and the linear scaling with system size, enables the *factor-solve* method to have considerably lower computation time than the reference method.

References

[1] H. Jóhannsson, A. H. Nielsen, and J. Østergaard, "Wide-Area Assessment of Aperiodic Small Signal Rotor Angle Stability in Real-Time," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4545–4557, 2013.

[2] T. Weckesser, H. Jóhannsson, and J. Østergaard, "Real-Time Remedial Action Against Aperiodic Small Signal Rotor Angle Instability," *IEEE Transactions on Power Systems*, vol. 31, no. 1, pp. 387–396, 2016.

[3] I. Šmon, G. Verbič, and F. Gubina, "Local voltage-stability index using Tellegen's theorem," *IEEE Transactions on Power Systems*, vol. 21, no. 3, pp. 1267–1275, 2006.

[4] S. Corsi and G. Taranto, "A Real-Time Voltage Instability Identification Algorithm Based on Local Phasor Measurements," *IEEE Transactions on Power Systems*, vol. 23, no. 3, pp. 1271–1279, 2008.

[5] Y. Wang, I. R. Pordanjani, W. Li, W. Xu, T. Chen, E. Vaahedi, and J. Gurney, "Voltage stability monitoring based on the concept of coupled single-port circuit," *IEEE Transactions on Power Systems*, vol. 26, no. 4, pp. 2154–2163, 2011.

[6] T. Weckesser, H. Jóhannsson, J. Østergaard, and T. Van Cutsem, "Sensitivity based assessment of transient voltage sags caused by rotor swings," in *Proceedings of the 18th Power Systems Computation Conference (PSCC)*, Wroclaw, Poland, 2014.

[7] J. G. Møller, H. Jóhannsson, and J. Østergaard, "Thevenin equivalent method for dynamic contingency assessment," in *Proceedings of IEEE Power & Energy Society's General Meeting*, Denver, CO, USA, 2015.

[8] ——, "Super-Positioning of Voltage Sources for Fast Assessment of Wide-Area Thévenin Equivalents," *IEEE Transactions on Smart Grid*, vol. 8, no. 3, pp. 1488–1493, 2017.

[9] P. Aristidou, D. Fabozzi, and T. Van Cutsem, "Dynamic Simulation of Large-Scale Power Systems Using a Parallel Schur-Complement-Based Decomposition Method," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 10, pp. 2561–2570, 2014.

[10] S. Sommer and H. Jóhannsson, "Real-time thevenin impedance computation," in *Proceedings of IEEE PES Innovative Smart Grid Technologies Conference (ISGT)*, Washington, DC, USA, 2013, pp. 1–6.

[11] S. Sommer, A. Aabrandt, and H. Jóhannsson, "Reduce-Factor-Solve for Fast Thevenin Impedance Computation and Network," *IET Generation, Transmission & Distribution*, nov 2018.

[12] H. Yuan and F. Li, "A comparative study of measurement-based Thevenin equivalents identification methods," in *2014 North American Power Symposium (NAPS)*. Pullman, WA, USA: IEEE, 2014, pp. 1–6.

[13] L. Giraud, A. Haidar, and Y. Saad, "Sparse approximations of the Schur complement for parallel algebraic hybrid solvers in 3D," *Numerical Mathematics-theory Methods and Applications*, vol. 3, no. 3, pp. 276–294, 2010.

[14] C. Hildebrandt, B. C. Karatas, J. G. Møller, and H. Jóhannsson, "Evaluation of Factorization Methods for Thévenin Equivalent Computations in Real-Time Stability Assessment," in *Proceedings of 20th Power Systems Computation Conference (PSCC)*, Dublin, Ireland, 2018.

[15] T. A. Davis, "Algorithm 907 : KLU , A Direct Sparse Solver for Circuit Simulation Problems," *ACM Transactions on Mathematical Software*, vol. 37, no. 3, pp. 1–17, 2010.

[16] ——, *Direct Methods For Sparse Linear Systems*, N. J. Higham, Ed. Gainesville, Florida: SIAM, 2006.

[17] H. Jóhannsson, J. Østergaard, and A. H. Nielsen, "Identification of critical transmission limits in injection impedance plane," *International Journal of Electrical Power & Energy Systems*, vol. 43, no. 1, pp. 433–443, 2012.

[18] F. Dorfler and F. Bullo, "Kron Reduction of Graphs With Applications to Electrical Networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 1, pp. 150–163, 2013.

[19] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education," *IEEE Transactions on Power Systems*, vol. 26, no. 1, pp. 12–19, 2011.

[20] CIGRÉ TF38.02.08, "Long Term Dynamics Phase II, Final Report," Tech. Rep., 1993.

[21] IEEE Standards Association, "C37.118.1-2011 IEEE Standard for Synchrophasor Measurements for Power Systems." Tech. Rep., 2011.

[22] M. Glavic and T. Van Cutsem, "Wide-Area Detection of Voltage Instability From Synchronized Phasor Measurements. Part I: Principle," *IEEE Transactions on Power Systems*, vol. 24, no. 3, pp. 1408–1416, 2009.

[23] ——, "Wide-Area Detection of Voltage Instability From Synchronized Phasor Measurements. Part II: Simulation Results," *IEEE Transactions on Power Systems*, vol. 24, no. 3, pp. 1417–1425, 2009.

[24] H. Jóhannsson, H. Morais, A. H. B. Pedersen, Q. Wu, and D. Ouellette, "SW-platform for R&D in Applications of Synchrophasor Measurements for Wide-Area Assessment, Control and Visualization in Real-Time," *CIGRE US National Committee 2014 Grid of the Future Symposium*, 2014.

Christina Hildebrandt Lüthje Jørgensen (S'17) received the M.Sc. degree in mathematical modelling and computation from the Technical University of Denmark in 2015, where she is currently pursuing the Ph.D. degree with the Centre of Electric Power and Energy, Department of Electrical Engineering. Her research interest includes developing high performance algorithms for assessing stability of power systems.

Jakob Glarbo Møller (M'17) received the M.Sc. and Ph.D. degrees in electrical engineering from the Technical University of Denmark in 2013 and 2017 respectively, where he is currently a postdoc with the Centre of Electric Power and Energy, Department of Electrical Engineering. His research interest includes algorithms for assessing operational security of power systems.

Stefan Sommer received his M.Sc. in mathematics in 2008 and his PhD in computer science in 2012 from the University of Copenhagen. He is currently Associate Professor at the Department of Computer Science, University of Copenhagen. His research interests cover aspects of mathematical modelling, numerical algorithms and statistics, including foundational and algorithmic problems in analysis of data with complex structure.

Hjörtur Jóhannsson (M'11) received the M.Sc. and PhD degrees in electrical engineering from the Technical University of Denmark in 2007 and 2011 respectively, where he is currently a Senior Scientific Consultant at the Center of Electric Power and Energy, Department of Electrical Engineering. His research interests concern the development of methods for secure and stable operation of power systems with a high share of RES based production, with special focus on real-time approaches.