SIAM J. MATRIX ANAL. APPL.
Vol. 26, No. 4, pp. 1150–1178

AN ITERATIVE METHOD FOR SOLVING COMPLEX-SYMMETRIC SYSTEMS ARISING IN ELECTRICAL POWER MODELING*

VICTORIA E. HOWLE† AND STEPHEN A. VAVASIS‡

Abstract. We propose an iterative method for solving a complex-symmetric linear system arising in electric power networks. Our method extends Gremban, Miller, and Zagha's [in *Proceedings of the International Parallel Processing Symposium*, IEEE Computer Society, Los Alamitos, CA, 1995] support-tree preconditioner to handle complex weights and vastly different admittances. Our underlying iteration is a modification to transpose-free QMR [6] to enhance accuracy. Computational results are described.

Key words. iterative methods, preconditioning, complex-symmetric systems, support-tree preconditioning, electrical power networks

AMS subject classifications. 65F10, 65F50

DOI. 10.1137/S0895479800370871

1. AC power networks.

Consider the linear system

$$A^T D^{-1} A\mathbf{v} = A^T D^{-1}\mathbf{b} \tag{1}$$

in which A is an $m \times n$ real matrix, D is an $m \times m$ complex diagonal matrix whose diagonal entries have positive real parts, $\mathbf{b}$ is a complex m-vector, and $\mathbf{v}$ is the n-vector of unknowns.

Equation (1) arises in the analysis of an alternating-current (AC) electrical network composed of generators and loads joined by a graph. Each node in the graph has a voltage, which is a complex number. The magnitude of the complex number is the magnitude of the voltage, and the argument is the phase difference of the voltage with respect to some reference phase.

Similarly, currents in the system are also complex numbers associated with graph edges. The generators can be modeled as voltage sources with a fixed voltage. The loads can be modeled as devices with fixed impedance. The impedance is a complex number with a positive real part.

If one is given the voltages of the generators and the impedances of the loads, then the problem of recovering the voltages at all nodes reduces to solving linear equations of the form (1). In this case, A is the *node-arc incidence matrix* (NAI) of the network. An NAI of a directed graph has one row for every edge of the graph and one column for every node. In each row, all entries are zeros except for exactly one "1" and one "−1" per row, which correspond to the endpoints of the graph edges. The diagonal

*Received by the editors April 10, 2000; accepted for publication (in revised form) by R. Freund July 26, 2004; published electronically May 6, 2005. This work was supported in part by NSF grant CCR-9619489, NSF grant DMS-9505155, and ONR grant N00014-96-1-0050. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.
http://www.siam.org/journals/simax/26-4/37087.html
†Sandia National Laboratories, P.O. Box 969, MS 9159, Livermore, CA 94551 (vehowle@sandia.gov).
‡Department of Computer Science, Cornell University, Ithaca, New York 14853 (vavasis@cs.cornell.edu).

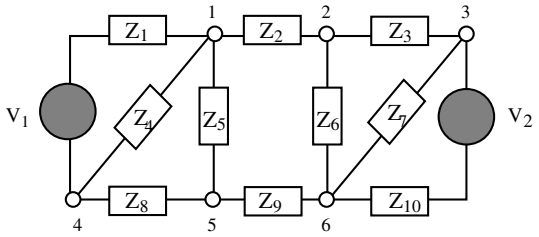

FIG. 1. *A simple AC network with two generators. Each edge j has a given constant impedance Z_j.*

matrix D stores the impedances of the loads, $\mathbf{b}$ holds the generator voltages, and $\mathbf{v}$ is the vector of node voltages. The linear system (1) is obtained from Ohm's law and Kirchhoff's law (current balance):

$$\begin{aligned} D\mathbf{i} + A\mathbf{v} &= \mathbf{b} \quad \text{(Ohm)}, \\ A^T\mathbf{i} &= \mathbf{0} \quad \text{(Kirchhoff)}. \end{aligned} \tag{2}$$

If we multiply Ohm's law by $A^T D^{-1}$ and apply current balance, we obtain the linear system (1).

As an example of the various components in (1), consider the simple network in Figure 1.

For this network,

$$A = \begin{bmatrix} 1 & & & -1 & & \\ 1 & -1 & & & & \\ & 1 & -1 & & & \\ 1 & & & -1 & & \\ 1 & & & & -1 & \\ & 1 & & & & -1 \\ & & 1 & & & -1 \\ & & & 1 & -1 & \\ & & & & 1 & -1 \\ & & 1 & & & -1 \end{bmatrix},$$

$$D = \operatorname{diag}(\begin{bmatrix} Z_1 & Z_2 & Z_3 & Z_4 & Z_5 & Z_6 & Z_7 & Z_8 & Z_9 & Z_{10} \end{bmatrix}),$$

$$\mathbf{b} = \begin{bmatrix} V_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & V_2 \end{bmatrix}^T. \tag{3}$$

When there are no faults, the diagonal elements of D, i.e., the impedances, are of approximately the same magnitude. When there is a fault in the network, e.g., a nearly open circuit exists in transmission lines, some of the impedances are much larger than the impedances associated with the functioning edges, making D extremely ill conditioned. See Bergen [1] for more information about modeling AC networks.

In the network shown in Figure 2, there are nearly open circuits in the edges associated with impedances Z_2 and Z_9. For this system, the matrix A and the vector $\mathbf{b}$ would be the same as those for the system in Figure 1, but the matrix D would now contain impedances of greatly varying magnitudes, e.g.,

$$D = \operatorname{diag}(\begin{bmatrix} Z_1 & \boldsymbol{Z_2} & Z_3 & Z_4 & Z_5 & Z_6 & Z_7 & Z_8 & \boldsymbol{Z_9} & Z_{10} \end{bmatrix}), \tag{4}$$

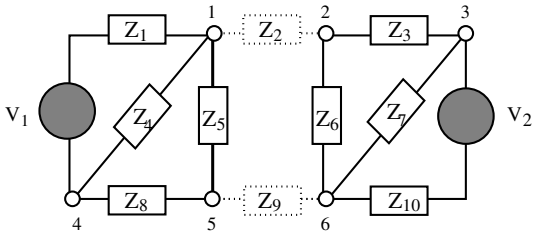

FIG. 2. *A simple AC network with two generators. Each edge j has a given constant impedance Z_j. In this example, we show faults in the edges associated with impedances Z_2 and Z_9. In the case of a nearly open circuit, for example, the magnitudes of Z_2 and Z_9 would be much greater than the magnitudes of the other impedances.*

where the $\boldsymbol{Z_2}$ and $\boldsymbol{Z_9}$ values are much larger in magnitude than the other impedances. Modeling faulty networks is important in practice since load-regulating devices must be designed to function properly even if part of the network fails.

We assume throughout this paper that the gap between the magnitudes of high impedance wires and low impedance wires is large and that removal of the high impedance wires disconnects the graph. In this case, the matrix $K = A^T D^{-1} A$ can be arbitrarily ill conditioned. It is not required that the removal of high impedance wires disconnect the graph for our algorithm to work; however, it is for this case that our method is a significant improvement over previous work [9].

We make two main contributions in this paper. The first contribution is an extension of Gremban's support tree preconditioner to cover complex weights (i.e., AC networks) and widely varying edge weights (i.e., faults). Even once we have a good preconditioner M, in the presence of a fault that disconnects the graph, K and therefore M^{-1} can be extremely ill conditioned separately. Even though the product $M^{-1}K$ is well conditioned, $M^{-1}(K\mathbf{v})$ may be computed inaccurately. Our second contribution is a technique that computes $M^{-1}(K\mathbf{v})$ accurately by splitting $K\mathbf{v}$ into its components in the range and null space of the functioning edges. For our algorithm to work efficiently, we need an efficient projection into the range and null spaces of the functioning edges.

A more general approach to achieving high accuracy in this kind of layered system was proposed by Bobrovnikova and Vavasis [3]. The method of [3] does not assume that there is an efficient projection into the range and null space of the functioning edges. But that method appears to be very difficult to precondition.

A direct algorithm known as complete orthogonal decomposition was proposed by Hough and Vavasis [13]. This method applies to the weighted least squares problem associated with faulted DC power networks. However, the method relies on the system being real and positive definite. There is no simple extension to the complex-symmetric case.

Other previous related work includes another combinatorial preconditioner for weighted node-arc adjacency matrices by Guo and Skeel [11], previous versions of support-tree preconditioners by Vaidya [18] and Bern et al. [2], and work by Vuik, Segal, and Meijerink [20] on a related mathematical problem arising in diffusion modeling using an explicit eigenvector projection. The problem analyzed by Vuik, Segal, and Meijerink involves a real, symmetric, positive definite matrix that is highly ill conditioned due to a large contrast in permeability coefficients in the system being modeled. The method proposed by Vuik, Segal, and Meijerink relies on a good choice

of projection vector, which involves knowing properties of the eigenvectors.

From the electrical power modeling perspective, there has been some work on using iterative methods for solving the complex-symmetric systems arising in electrical power modeling; see, e.g., [4], [7], [15], [17]. However, these methods have not addressed the ill-conditioning associated with faults in the electrical power system.

2. Support trees. Our system (1) is singular because the nodes are ungrounded. Since the voltage values are potentials, if we have not set one of the nodes to some reference voltage (i.e., grounded the node), we can add an arbitrary constant to all of the voltages and have an equally valid solution. Mathematically, this means that the matrix $A^T D^{-1} A$ is singular, because the vector of all 1's is in the nullspace of A. We address this detail by projecting vectors onto the range space of $A^T D^{-1} A$. This projection is ignored for the remainder of the paper. After grounding the system, it is still ill conditioned for two reasons.

The first source of ill-conditioning is inherent in NAI matrices. For example, if A describes an $n \times n$ grid-graph, $\kappa(A^T A) = O(n^2)$. This ill-conditioning has been addressed by *support-tree* preconditioners developed by Gremban, Miller, and Zagha [10]. We also use support-tree preconditioners in our method. The second source of ill-conditioning is caused by the widely varying weights in the faulted system. Gremban, Miller, and Zagha analyze only the case of nearly equal weights. We extend their analysis of condition numbers of the preconditioned system first to subgraphs having edge weights with widely varying magnitudes and then to graphs with complex edge weights.

2.1. Support trees of Gremban, Miller, and Zagha. Gremban, Miller, and Zagha form a support-tree preconditioner as follows. First, divide the nodes of the network graph into some number of approximately equal-sized subgraphs. Then recursively subdivide the subgraphs, etc., until all of the individual nodes have been separated. Note that the method does not depend on the number of subgraphs in each subdivision. For the results in this paper, we recursively subdivide into quarters. Next, build a tree based on this partitioning. The root of the tree is in correspondence with the entire original graph. The children of the root are in correspondence with the subgraphs of the graph obtained from the first partition, and so on down to the leaves of the tree, which are in correspondence with the individual nodes of the original graph. (See Figure 3.)

Next assign weights to the edges of this tree based on the edge weights in the original graph. Let G be the original network graph and let S be the support tree. Let v be a support-tree node corresponding to subgraph V of G, and let e be the support-tree edge from v to its parent. Assign weight to e equal to the sum of the conductances (i.e., reciprocal resistances) of edges in G connecting V to $G - V$. In other words, the weight on e is the sum of the entries of D^{-1} corresponding to the nodes of V.

DEFINITION 1. *We define the* weighted Laplacian matrix $L(G)$ *of an* n*-node graph* G *as the* $n \times n$ *matrix whose* j*th diagonal entry corresponds to the sum of weights of the edges incident on the* j*th node of the graph. The* (i, j) *entry of* $L(G)$ *is equal to the negative of the weight of the edge in* G *connecting nodes* i *and* j*.*

Note that the system matrix K is the weighted Laplacian of the input graph G. Let T be the weighted Laplacian matrix of the new network S. If n is the number of original circuit nodes and t is number of nonsingleton subgraphs created during partitioning, then $n + t$ is the number of support-tree nodes. The matrix T is an $(n + t) \times (n + t)$ very sparse matrix.

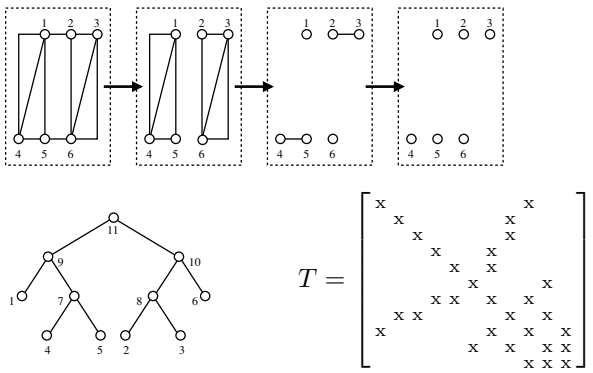

FIG. 3. *Example of constructing a support tree. The top left picture shows a network graph with six nodes and ten edges. Successive cuts into subgraphs are shown in the pictures progressing left to right. The support tree is shown on the bottom left, with the corresponding Laplacian matrix T on the bottom right. Notice that the leaves of the support tree correspond to the nodes of the network graph, and the root of the support tree corresponds to the entire original network graph.*

Let M be the Schur complement of T obtained by eliminating the internal tree nodes, that is, the tree nodes that are not leaves. Gremban, Miller, and Zagha showed that M is a good preconditioner for $A^T D^{-1} A$ in the real case (DC) with uniform edge weights (no faults). In particular, they showed that for equally weighted instances of (1) (i.e., $D = I$), the condition number of grid-graphs is reduced from $O(n^2)$ to $O(n \log n)$. Although M is dense, linear systems of the form $M\mathbf{v} = \mathbf{r}$ can nonetheless be solved in linear time using Cholesky factorization on the larger sparse matrix T. Note that T has a perfect elimination order since it is the weighted Laplacian matrix of a tree and we can eliminate from the leaves to the root. They show that solving

$$\begin{pmatrix} T_1 & T_2 \\ T_2^T & T_3 \end{pmatrix} \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{r} \\ \mathbf{0} \end{pmatrix} \tag{5}$$

and letting $\mathbf{v} = \mathbf{v}_1$ are equivalent to solving $M\mathbf{v} = \mathbf{r}$, where $M = T_1 - T_2 T_3^{-1} T_2^T$ is the support-tree preconditioner. Thus, the preconditioner is efficient in practice.

2.2. Extensions of the support-tree preconditioner. To extend the idea of support-tree preconditioners to the case of an AC network with faults, we change how we build the support tree as follows. First, assume that removal of the faulted edges disconnects the graph into at least two subgraphs. (If this assumption does not hold, then our method does not constitute an improvement over previous work [9].) We require the top level of the support tree to be composed entirely of faulted edges, and children of the root should be connected subgraphs of the network after faulty edges are deleted. We include all of the faulted edges in the first separator. This in turn puts all of the weights corresponding to faulted edges in the top level of the support tree. We build the rest of the support tree as before. We show below that with this change, the support-tree preconditioners are good preconditioners for $A^T D^{-1} A$. In applying the preconditioner M in the AC case, the matrices $K = A^T D^{-1} A$ and M are complex-symmetric and hence the Cholesky factorization technique does not apply directly. We show that the LU factorization can still be stably performed

without pivoting (and thus without fill-in). Thus, as before, we can solve systems of the form $M\mathbf{v} = \mathbf{r}$ by solving the larger sparse system involving T.

THEOREM 2. *T has an LU factorization, and the elements of L are bounded; i.e., the computation is stable without pivoting.*

Proof. The diagonal elements of T are exactly the negative sums of the off-diagonal elements. In addition, because we have ordered the tree nodes from the leaves to the root, T has perfect elimination order. The elements of L are thus only 0's, 1's, and -1's. The L matrix has 1's on the diagonal, -1's in the elements of T below the diagonal that are nonzero, and 0's elsewhere. ∎

In addition to the ill-conditioning inherent in NAI matrices, there is also ill-conditioning in our application of (1) caused by widely varying weights in a faulted system. Let ρ_1 be the absolute value of a typical admittance in functioning wires, and let ρ_2 be the absolute value of a typical admittance in faulty wires. Then $\rho_1 \gg \rho_2$, and the original linear system can be decomposed as $K = A^T D^{-1} A = K_1 + K_2$, where $K_1 = A_1^T D_1^{-1} A_1$ and $K_2 = A_2^T D_2^{-1} A_2$. Here subscripts $1, 2$ denote the partition into functioning and faulty wires, respectively; hence $\|D_1\| \approx \rho_1$ and $\|D_2\| \approx \rho_2$.

Gremban, Miller, and Zagha analyze only the case of nearly equal weights. We extend their analysis of condition numbers of the preconditioned system first to graphs having edge weights with widely varying magnitudes, assuming the change in forming the support tree that we discussed above and then to graphs with complex edge weights. We assume for simplicity in the following theorems that the faults are nearly open circuits, i.e., low admittance wires.

2.3. Extensions to analysis. In the rest of this section, we extend Gremban's analysis as follows. Since electrical power networks are laid out geographically, $n \times n$ grid-graphs are a reasonable first approximation to consider. Therefore, we first extend Gremban's analysis to DC networks (i.e., networks in which the edges have real weights) with faults by proving that in the DC grid-graph case with faults along the median edges, the condition number of the preconditioned system is $O(n \log n)$, where n^2 is the number of nodes in the system. The median edges are the edges running through the middle of the grid-graph, horizontally and vertically. That is, removal of the median edges would divide the grid-graph into four approximately equal-sized subgraphs. Since in general electrical power networks are not actually laid out on grid-graphs and faults are not confined to being along the median edges, we show results for general graphs. Little is known about the behavior of support-tree preconditioners on general graphs. Although we do not have an upper bound on the condition number of the preconditioned system on a general graph, we can show that for general DC networks with arbitrarily located faults, there is a bound on the condition number of the preconditioned system that is independent of the relative magnitude of the faults. Finally, we extend our results for grid-graphs and general graphs to the AC case. Given certain assumptions about the impedance values in the network, we show that we can bound the condition number in the AC faulted case based on the condition number of the related DC network obtained by taking the real part of the impedance values. In particular, we assume that the impedance values in the network (the diagonal elements of D) lie in a pointed cone in the complex plane; i.e., if d_j is a diagonal element of D, then $d_j = x_j + iy_j$ where $x_j > 0$ and $|y_j| \leq \mu x_j$ for some positive cone constant μ. This series of theorems then shows that we can extend Gremban's support-tree preconditioner, with certain changes, to be a good preconditioner in the case of AC networks with faults.

2.3.1. Extensions to analysis of DC grid-graph networks with faults. We first extend Gremban's analysis to the case of DC (real) networks with faults. We first introduce some notation that will be useful in the proofs that follow. Note that the definitions that do not specifically rely on the matrices being real apply equally to the AC case.

DEFINITION 3. *We refer to the* condition number *of the preconditioned system* $M^{-1}K$ *as* $\kappa(M,K)$*, where*

$$\kappa(M,K) = \max \frac{|\mathbf{x}^* M \mathbf{x}|}{|\mathbf{x}^* K \mathbf{x}|} \cdot \max \frac{|\mathbf{x}^* K \mathbf{x}|}{|\mathbf{x}^* M \mathbf{x}|}, \tag{6}$$

where the maxima are taken over nonzero vectors $\mathbf{x}$ *in the range space of* K *(which is equal to the range space of* M*). In the real case, the absolute value symbols are not needed.*

Note that this definition is equivalent to the usual eigenvalue definition of condition number in the real case.

We next prove a series of lemmas that we will use to show that in the DC (real) case, the upper bound on the condition number of the preconditioned system does not depend on the values of the faulted edges, and that for $n \times n$ (real) grid-graphs, the condition number is $O(n \log n)$. The key technique we will use is support numbers.

DEFINITION 4 (see [9]). *For two real positive semidefinite matrices* A *and* B*, the support of* B *for* A*,* $\sigma(A,B)$*, is defined to be the greatest lower bound over all* τ *such that* $\tau B - A$ *is positive semidefinite.*

Gremban relates this quantity to the condition number as follows.

LEMMA 5. *Let* A, B *be real positive semidefinite matrices. Then* $\kappa(A,B) = \sigma(A,B)\sigma(B,A)$.

Thus, for our grid-graph construction, we must obtain upper bounds on $\sigma(M,K)$ and $\sigma(K,M)$.

We start with $\sigma(M,K)$. The support tree for the grid-graph is not completely regular because subgraphs that are adjacent to the boundary of the entire grid have fewer edges emanating from them than subgraphs on the interior. Our analysis of $\sigma(M,K)$ is simplified by assuming, however, that all subgrids of size $2^h \times 2^h$ have exactly $4h$ edges emanating from them. This assumption may be made without loss of generality for the following reason. Let T' be the network resulting from augmentation of T with these extra edges. Then $T' - T$ is positive semidefinite (since inserting edges corresponds to adding a semidefinite matrix); hence so is $M' - M$, where M' is the Schur complement of T' according to the following theorem.

THEOREM 6 (see [12]). *Let* A, B *be* $n \times n$ *symmetric positive semidefinite matrices of the same size, and let* k *be an integer between* 1 *and* $n-1$*. Assume that the upper left* $k \times k$ *submatrices of both* A *and* B *are invertible, and let* $\mathrm{Schur}_k(A)$ *denote the Schur complement of the upper left* $k \times k$ *submatrix (i.e.,* $\mathrm{Schur}_k(A) = A(k+1:n, k+1:n) - A(k+1:n, 1:k)A(1:k, 1:k)^{-1}A(1:k, k+1:n)$*). Similarly, let* $\mathrm{Schur}_k(B)$ *be the Schur complement of the upper left* $k \times k$ *submatrix of* B*. Then if* $A - B$ *is positive semidefinite, so is* $\mathrm{Schur}_k(A) - \mathrm{Schur}_k(B)$.

This means that if τ is a scalar such that $\tau K - M'$ is positive semidefinite, so is $\tau K - M' + (M' - M) = \tau K - M$. Therefore, $\sigma(M,K) \leq \sigma(M',K)$, so any upper bound for $\sigma(M',K)$ applies also to $\sigma(M,K)$.

Hence we assume T has the regular structure mentioned earlier for the remainder of this analysis. The next step in the analysis of $\sigma(M,K)$ is to obtain upper bounds for the off-diagonal entries of M. This is the purpose of the next three lemmas.

LEMMA 7. *Let M be the Schur complement of T as described earlier. Then $M(j,k)$ is the kth entry of $\mathbf{i}$, where $\mathbf{i}$ and $\mathbf{y}$ satisfy the following equation in which $\mathbf{e}_j$ denotes the jth column of the identity matrix:*

$$T \begin{bmatrix} \mathbf{y} \\ \mathbf{e}_j \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{i} \end{bmatrix}. \tag{7}$$

Remark 1. This lemma has an interpretation in terms of electrical networks. Recall that multiplying a weighted Laplacian matrix of a graph by a vector $\mathbf{v}$ corresponds to assigning voltages to the nodes given by $\mathbf{v}$ and then determining the excess currents at the nodes. Therefore, the above lemma corresponds to holding leaf node j of the support tree to voltage equal to 1, the other leaf nodes to voltage 0, letting the nonleaf tree nodes "float" (i.e., assume whatever voltage is needed so that the current at the node balances), and measuring the excess current at node k.

Proof. This lemma holds because M is obtained from T by performing Gaussian elimination steps on (7) to eliminate the nonleaf tree nodes. If we perform Gaussian elimination on (7), we obtain $M\mathbf{e}_j = \mathbf{i}$; i.e., the kth entry of $\mathbf{i}$ is equal to $M(j,k)$. □

There is exactly one path between nodes j and k in the support tree. In estimating $\mathbf{i}(k)$, we need to consider two cases: either the path between nodes j and k goes through the root node (Lemma 8), or it does not (Lemma 9).

LEMMA 8. *Let M be the support-tree preconditioner for the $n \times n$ grid-graph with edge weights described above. Assume n is an exact power of 2. Let l be the number of levels in the tree, i.e., $n = 2^l$, assuming exact quadrisection at each level. Let the median (faulted) edges have admittance $\epsilon/4$ and the nonfaulted edges have admittance $1/4$. Assume the path from j to k passes through the root node of the support tree. Then*

$$|M(j,k)| \leq \frac{49 \cdot 2^{-3l}\epsilon}{288} + O(2^{-6l}\epsilon) + O(\epsilon^2).$$

Remark 2. The assumption about powers of 2 is made to simplify the proof and the notation. The factor of $1/4$ unclutters the figures but is otherwise unnecessary.

Proof. Although this lemma can be proved using purely algebraic arguments, we prefer to argue using principles of electrical networks because we believe this gives more insight.

The first part of the proof contracts T almost to a path using series-parallel equivalent circuits. A similar argument was used by Gremban. For example, consider two leaf nodes of T, both holding a voltage 0, attached to the same parent node p with edges that both have resistance r. These two leaf nodes can be merged into a single node of voltage 0 connected to the same parent with resistance $r/2$. Then, since p is a floating node and is connected to exactly two edges with resistance r_1 and r_2, p can be deleted and the two edges can be replaced by a single edge with resistance $r_1 + r_2$. Proceeding in this manner, we can merge and contract many tree nodes as shown in Figure 4.

As in Gremban's analysis, if we reduce nodes up to a child w of a node u at level $(i-1)$, the reduced system is equivalent to a node connected to u with an edge resistance less than or equal to $1/2^{i-1}$. (It can be shown using electrical reasoning or algebra that overestimating the resistance of an off-path edge will lead to overestimation of $|M(j,k)|$, which is valid since we are trying to obtain an upper bound on this quantity.)

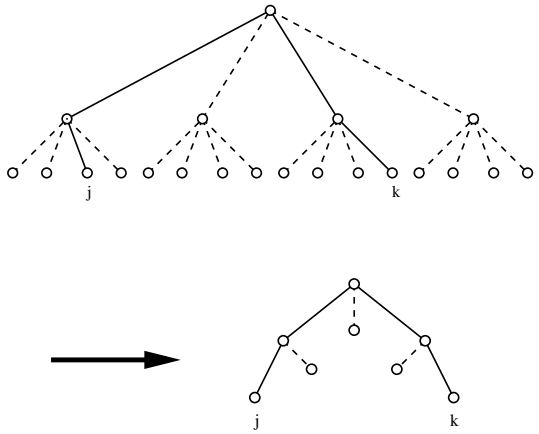

FIG. 4. *The solid lines in the first tree show the path between nodes j and k. The dashed lines are the edges that we reduce up to the (j,k)-path. The second tree shows the result of reducing all other nodes up to the path between j and k. The (j,k)-path is again shown with solid lines, with the reduced edges shown as dashed lines.*

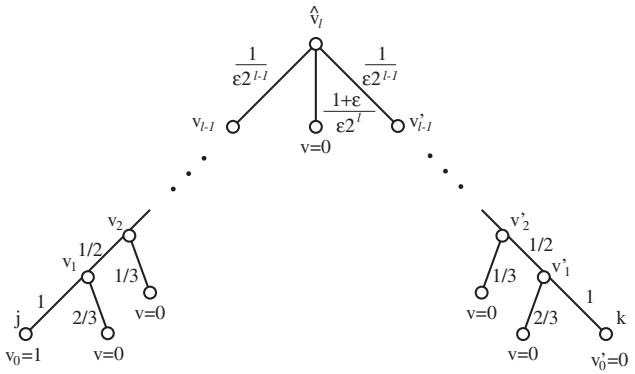

FIG. 5. *Support-tree reduced to the (j,k)-path. The node $\hat{v}'_l$ represents the root of the tree, v_0 represents the j-node, and v'_0 represents the k-node. The edge labels shown are resistances.*

Therefore, the reduced system will look like the system in Figure 5 for a power-of-two grid. In this figure, v_0 represents the jth node of the tree (whose voltage has been set to one), v'_0 represents the kth node of the tree (whose voltage has been set to zero), and $\hat{v}_l$ represents the root of the tree.

Once we have reduced the system to the (j,k)-path, we solve for the net current at the kth node, i.e., at v'_0 in Figure 5. Recall that finding this current is equivalent to finding $\mathbf{i}(k) = M(j,k)$.

Solving current balance equations at each floating node, we get the recurrence relations

$$4v_{i+2} - 9v_{i+1} + 2v_i = 0, \tag{8}$$
$$4v'_{i+2} - 9v'_{i+1} + 2v'_i = 0.$$

We also have equations for the root node and its neighbors. Using standard

techniques, we can solve these recurrences to obtain

$$\mathbf{i}_k = \frac{49\epsilon 2^{-3l}}{288} + O(\epsilon 2^{-6l}) + O(\epsilon^2), \tag{9}$$

which bounds the element $M(j,k)$ in the case where the (j,k)-path goes through the root of the support tree. For details of the analysis, see [14]. ∎

Next we deal with the case where the (j,k)-path does not pass through the root of the tree.

LEMMA 9. *Let M be the support-tree preconditioner as defined above. Then $|M(j,k)| \le 3.5/8^h$ assuming the (j,k)-path does not pass through the root of the tree. Here h is the height of the common ancestor of j,k in the support tree.*

Proof. As in Lemma 8, we begin by reducing up to the (j,k)-path. In this case, after reducing the tree as before to the (j,k)-path, we have a completely unfaulted path between nodes j and k in the support tree, with an extra path off of the root $\hat{v}_l$ containing all of the faulted edges and other edges hanging off the path. The same recursion (8) applies to this analysis, and the same techniques can be used to solve it. ∎

The approach used for estimating $\sigma(M,K)$ is the same as Gremban's; namely, we first partition the graph and the preconditioner, and then for each piece we apply a congestion/dilation argument. The first lemma concerns partitioning.

LEMMA 10. *If $A = A_1 + \cdots + A_s$ and $B = B_1 + \cdots + B_s$, where each A_i and each B_i is symmetric positive semidefinite, then*

$$\sigma(A,B) \le \max\{\sigma(A_1,B_1), \ldots, \sigma(A_s,B_s)\}.$$

Proof. This lemma is proved by Gremban [9, Chapter 4]. ∎

The particular partitioning to be used in our analysis is as follows. As in Gremban, we write $K = K_1 + \cdots + K_l$ and $M = M_1 + \cdots + M_l$. The partition of K is given by $K_h = 2^{h-l-1}K$. Thus, $K_l = K/2$, $K_{l-1} = K/4$, etc. (This leaves the fraction 2^{-l} of K "unused." This is valid because underestimating K can only increase $\sigma(M,K)$.) The partition of M is given by the rule that $M_h(i,j) = M(i,j)$ provided that the highest tree node in the support tree on the (i,j)-path is at height h; else $M_h(i,j) = 0$. (We order "height" so that the leaves are at height 0 and the root at height l.)

The next part of the analysis of $\sigma(M,K)$ involves a congestion-dilation argument. The terms are defined as follows.

DEFINITION 11 (see [9]). *Let A and B be graphs. Let p_0 be an injective mapping from nodes of A into nodes of B. Suppose there exists a mapping p from edges in A to paths in B such that if $e = (a,b)$ is an edge in A,then the endpoints of $p(e)$ are $(p_0(a), p_0(b))$. For an edge f in B, let $d_1, \ldots, d_k$ be the edges in A such that $f \in p(d_i)$; that is, d_i are the A edges whose embedding path in B includes edge f. The* congestion *of edge f is*

$$\gamma(A,B,f) = \frac{\sum_{i=1}^{k} weight(d_i)}{weight(f)}. \tag{10}$$

The congestion *of the mapping $\gamma(A,B)$ is the maximum congestion over all edges in B.*

DEFINITION 12 (see [9]). *Given an embedding of a graph A into a graph B as in Definition* 11*, the* dilation *of the embedding is defined to be the length of the longest path in B onto which an edge of A is mapped.*

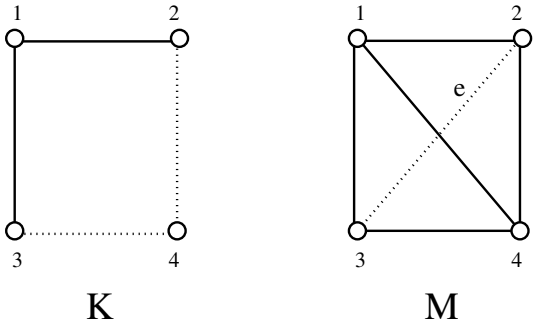

FIG. 6. *The graph K on the left is the original four-node system, and the graph M on the right is the graph corresponding to the Schur complement support-tree preconditioner M. Consider edge $e \in M$ (the dotted edge in the figure connecting M vertices $v_M = 2$ and $w_M = 3$). The corresponding path in K is the one shown by dotted edges in K (connecting K-nodes $v_K = 2$ to 4 to $w_K = 3$).*

LEMMA 13. *The support $\sigma(A, B)$ of a matrix B for a matrix A is less than or equal to the maximum congestion over edges in B times the maximum dilation over paths in B.*

Proof. This lemma is proved by Gremban [9, Chapter 4]. □

In order to apply the congestion–dilation analysis, we assume the following standard mapping of the edges of M onto paths in the grid-graph corresponding to K. Let e be an edge in M. The nodes of M correspond to the nodes in the original network K. If edge $e \in M$ connects nodes v_M and w_M in M, we map edge e to the path in K obtained by starting at the node $v_K \in K$ that corresponds to v_M and moving vertically until we reach the same row as node w_K (the node in K corresponding to node w_M). We then complete the path by moving horizontally until we reach node w_K.

As an example, consider the very simple four-node system shown in Figure 6. Form the support tree by bisecting the graph repeatedly, form the weighted Laplacian matrix T from the support tree, and then let M be the Schur complement obtained by performing Gaussian elimination on the internal (nonleaf) nodes of the tree. The graph corresponding to the preconditioner M is then as shown in the same figure. Consider edge $e \in M$ (the dotted edge in Figure 6 connecting M vertices $v_M = 2$ and $w_M = 3$). The corresponding path in K is the one shown by dotted edges in K (connecting K-nodes $v_K = 2$ to 4 to $w_K = 3$).

Now finally we have enough tools to estimate $\sigma(M, K)$.

LEMMA 14. *Let $K = A^T D^{-1} A$ correspond to a DC (real weights) network on an $n \times n$ grid-graph containing wires of two magnitudes, functioning wires with conductance $1/4$ and faulty wires with conductance $\epsilon/4$, where the faulty wires are along the median edges of the grid. Let M be the support-tree preconditioner as defined above. Then $\sigma(M, K) = O(n)$.*

Proof. As explained above, we estimate $\sigma(M_h, K_h)$ for each $h = 1, \ldots, l$ by considering $\gamma(M_h, K_h) \cdot \delta(M_h, K_h)$. Let us pick an h and select an e in K. We try to determine $\gamma(M_h, K_h, e)$. There are two cases to consider: either e is one of the median edges of K whose conductivity is $\epsilon/4$ or it is another edge.

Let us consider the median-edge case first. We observe that $\gamma(M_h, K_h, e) = 0$ if $h < l$ since no edge in M_h corresponds to a path that passes through the root node. This is because for $h < l$, all the edges in M_h correspond to K-paths that lie inside one of the four quadrants of K (after the median edges are removed). Thus, e does

not support any of these M-edges.

Thus, for a median edge we need consider only $\gamma(M_l, K_l, e)$. Let $d_1, \ldots, d_p$ be the edges in M_l whose corresponding K-paths $s_1, \ldots, s_p$ pass through e. Note that the tree paths corresponding to $d_1, \ldots, d_p$ must all pass through the root of the support tree. Therefore, the weight in M_l of each of these p edges is $(49\epsilon 2^{-3l})/288 + O(\epsilon 2^{-6l}) + O(\epsilon^2)$ by Lemma 8. Next, we have $p \le 2^{3l}$. This is because e is either in the same row (if e is horizontal) or column (if e is vertical) of one of the two endpoints of each s_i. Therefore, the number of possible endpoints for one end of s_i is n; the number of possible endpoints for the other end is n^2 (the total number of nodes). Thus, $p \le n^3 = 2^{3l}$. The weight in K_l of this edge e is $\epsilon/8$ (since $K_l = K/2$). Therefore, the maximum congestion in this case is

$$\begin{aligned}\gamma(M_l, K_l, e) &= \frac{\sum_{\mu=1}^{p} \frac{49\epsilon 2^{-3l}}{288} + O(\epsilon 2^{-6l}) + O(\epsilon^2)}{\text{weight}(e)} \\ &= \frac{O(2^{3l}\frac{49}{288}\epsilon 2^{-3l})}{\epsilon/8} \\ &= O(1).\end{aligned}$$

Next, consider the case that e is a functioning edge of K. Let (i, j) be the endpoints of e. Let h' be the height of the highest node z on the support-tree path connecting i to j. By assumption for this case, $h' < l$. If $h < h'$, then $\gamma(M_h, K_h, e) = 0$ because no paths induced by M_h-edges can use edge e, since any path in K induced by an edge of M_h uses nodes whose highest common ancestor in the support tree is at level h.

Thus, assume $h \ge h'$. Let $d_1, \ldots, d_p$ be the edges in M_h whose corresponding K-paths $s_1, \ldots, s_p$ pass through e. This means that $s_1, \ldots, s_p$ are all contained in a $2^h \times 2^h$ subgrid that also contains e, namely, the subgraph of K corresponding to the nodes in the support tree at z', where z' is the unique ancestor of z at height h. The total number p of such paths that can use e is therefore at most 8^h. This is because, depending on whether e is vertical or horizontal, one endpoint of each s_i is either in the same row or in the same column as e, so there are at most 2^h choices for one of the endpoints. The other endpoint could be anywhere in the subgrid determined by z', which has 2^{2h} nodes total.

Note that the weight of any M_h-edge is at most $3.5/8^h$ by Lemma 9. The weight of e in K_h is $(1/4) \cdot 2^{h-l-1} = 2^{h-l-3}$ by definition of K_h and the assumption that e is a functioning edge. Thus,

$$\begin{aligned}\gamma(M_h, K_h, e) &= \frac{\sum_{\mu=1}^{p} \text{weight}(d_\mu)}{\text{weight}(e)} \\ &= \frac{8^h \cdot 3.5/8^h}{2^{h-l-3}} \\ &= O(2^{l-h}).\end{aligned}$$

Thus, we have shown in all cases (both functioning and faulty wires) that for all e, $\gamma(M_h, K_h, e) = O(2^{l-h})$; hence $\gamma(M_h, K_h) = O(2^{l-h})$. Next, we observe that $\delta(M_h, K_h) = 2^h$. This is because the path in K_h induced by a node in M_h lies in the subgraph induced by a node in the support tree at height h, a grid whose diameter is 2^h. Thus, by Lemma 13, $\sigma(M_h, K_h) \le O(2^{l-h}) \cdot 2^h = O(2^l) = O(n)$. This holds for all h, so by Lemma 10, $\sigma(M, K) = O(n)$. ∎

Next, we turn our attention to $\sigma(K, M)$. The following lemma greatly simplifies this analysis.

LEMMA 15. *Let K, M be two $n \times n$ symmetric positive semidefinite matrices. Let M be the Schur complement of the lower right block of a larger symmetric positive semidefinite matrix T as in* (5)*. Let $\tilde{K}$ be K extended with zeros so that it is the same size as T; i.e.,*

$$\tilde{K} = \begin{bmatrix} K & 0 \\ 0 & 0 \end{bmatrix}. \tag{11}$$

Then $\sigma(K, M) \leq \sigma(\tilde{K}, T)$.

Proof. See Gremban [9, Chapter 4] for the proof. This also follows directly from Theorem 6 as follows. Define $\tilde{K}_\epsilon = [K, 0; 0, \epsilon I]$, where $\epsilon > 0$ and I is the identity matrix. Then clearly K is the Schur complement of the lower right block of $\tilde{K}_\epsilon$, so $\sigma(K, M) \leq \sigma(\tilde{K}_\epsilon, T)$ by the theorem. Then take the limit as $\epsilon \to 0$. ∎

The analysis of $\sigma(K, M)$ is now fairly straightforward.

LEMMA 16. *Let $K = A^T D^{-1} A$ correspond to a DC (real weights) network on an $n \times n$ grid-graph containing wires of two magnitudes—high admittance (functioning) wires and low admittance (nearly open circuit) wires—where the faults are along the median edges of the grid. Let T be as defined above and let $\tilde{K}$ be as defined in Lemma* 15*. Then $\sigma(\tilde{K}, T) = O(\log n)$.*

Proof. In this case, we are considering how well the support-tree matrix T supports the extended system matrix $\tilde{K}$. The only edges of $\tilde{K}$ are the edges of K; therefore we need only to map edges of K onto paths of T. We assume the following standard mapping of edges of K onto paths in T. Let e be an edge in K connecting nodes v_K and w_K. The leaves of the support tree correspond exactly to the nodes of the original network. Since v_K and w_K are nodes in the original network, they correspond to two leaf nodes in the support tree. We map edge e to the unique path between the corresponding leaves v_T and w_T in the support tree.

As before, the support of matrix T for $\tilde{K}$ is less than or equal to the product of the maximum congestion over edges in T and the maximum dilation over paths in T; i.e., $\sigma(\tilde{K}, T) \leq \gamma(\tilde{K}, T)\delta(\tilde{K}, T)$. We bound these two quantities separately.

The maximum dilation comes from the edge in K that must be mapped through a path that goes through the root of T. The length of this path is $2 \log_2 n + 2$.

Next we find the maximum congestion. Let p be the mapping described above from edges in K to paths in T, let e be an edge in T, and let $d_1, \ldots, d_k$ be the edges in K such that $e \in p(d_i)$ for some i.

Assume that edge $e \in T$ connects nodes v_j and v_k, where v_j is a child node of v_k. Then we have defined the edge weights of T such that the weight of edge e equals the sum of the edges in K that connect the nodes associated with the leaves of the tree rooted at v_j to the rest of the graph. This sum is exactly $\sum_{i=1}^{k} \text{weight}(d_i)$. Therefore, the congestion over any edge in T is one.

Thus, the support $\sigma(\tilde{K}, T)$ is less than or equal to $1 \times O(\log n) = O(\log n)$. ∎

Using Lemma 5 and Lemmas 13 through 16, we can show that in the case of an $n \times n$ grid-graph with faults along the median edges of the grid, the condition number of the preconditioned system is $O(n \log n)$. We summarize this result in the following theorem.

THEOREM 17. *Let $K = A^T D^{-1} A$ correspond to a DC (real weights) network on an $n \times n$ grid-graph containing wires of two magnitudes—high admittance (functioning) wires and low admittance (nearly open circuit) wires—where the faults are along the*

median edges of the grid. Let the preconditioner M be as above. Then $\kappa(M,K) = O(n \log n)$.

Proof. This follows from Lemmas 5, 14, and 16. ∎

2.3.2. Extensions to analysis of general DC networks with faults. In the nongrid case, we show that the condition number of the preconditioned system has a bound that is independent of the magnitude of the fault.

Since the condition number of the preconditioned system is bounded above by the product of the support of K for M, $\sigma(M,K)$, and the support of T for $\tilde{K}$, $\sigma(\tilde{K},T)$, we prove this result by showing that $\sigma(M,K)$ and $\sigma(\tilde{K},T)$ are each independent of the magnitude of the fault. We do this by considering the congestion and dilation of the respective embeddings. We show that both the congestion and dilation are independent of ρ_1 and ρ_2 for the case of K supporting M and for the case of T supporting $\tilde{K}$.

LEMMA 18. *Let $K = A^T D^{-1} A$ correspond to a connected DC (real weights) network containing wires of two magnitudes—high admittance (functioning) wires and low admittance (nearly open circuit) wires. Let the preconditioner M be as above. Assume that the admittance of a typical functioning edge is ρ_1, and the admittance of a typical faulted edge is ρ_2. Then $\sigma(M,K)$ has an upper bound that is independent of ρ_1 and ρ_2.*

Proof. In this lemma, we are considering the case of K supporting M, $\sigma(M,K)$. Fix a mapping from edges in M to paths in K. Such a mapping exists by the assumption that the network is connected. We further require that if there exists a path from a to b in K using unfaulted edges, then such a path must be selected for the embedding. The dilation is given by the longest path in K onto which an edge of M is embedded. For any mapping we choose, this path can be as long as the longest path in the K graph, but this length does not depend on ρ_1 or ρ_2.

For the congestion, we follow a proof similar to that for the grid-graph case. As in that case, we need an upper bound on the elements of M, which will give us a bound on the edge weights of the graph associated with M. Choose an edge $e \in K$. There are two cases to consider: either e is a faulted edge or it is not a faulted edge. If e is not a faulted edge, then weight$(e) = \rho_1$. Let $d_1, \ldots, d_k$ be the M edges whose K-paths include edge e. We claim that these edges all have weight of approximately ρ_1. We show this by using the same process of branch/path reduction as in Lemma 14, reducing the support tree to the (j,k)-path that connects the leaf nodes of T that correspond to the nodes in K connected by edge e. We can get an upper bound on the current at node k by removing the faulted branch of the reduced system. We can do this because the extra edges associated with the faulted parts of the system will only make the current that gets to the kth node less. The current at the kth node is therefore $O(\rho_1)$ giving us that $M(j,k) = O(\rho_1)$. As before, the current at node k equals the value of $M(j,k)$, so the M edges whose K-paths include edge e have weights of ρ_1. Thus the congestion in this case is approximately $O(\rho_1)/\rho_1 = O(1)$ and is independent of ρ_1 and ρ_2.

If e is a faulted edge, then weight$(e) = \rho_2$. We again let $d_1, \ldots, d_k$ be the M-edges whose K-paths include edge e. We claim that in this case these edges have weights approximately equal to ρ_2. We again show this by using the same process of branch/path reduction as in Lemma 14. By construction, any K-path using e must be a path between different K_1-subgraphs of K. Therefore, the (j,k)-path in T must go through the root of the tree. Contracting as before to the (j,k)-path, we can see that the current at node k must be $O(\rho_2)$. Therefore, $M(j,k) = O(\rho_2)$, and the M

edges whose K-paths include edge e have weights of $O(\rho_2)$. Thus the congestion in this case is approximately $O(\rho_2)/\rho_2 = O(1)$ and is independent of ρ_1 and ρ_2.

Since we have now shown that the dilation and congestion are separately bounded independently of ρ_1 and ρ_2, their product, which is an upper bound on the support of K for M, $\sigma(K, M)$, is also bounded independently of ρ_1 and ρ_2. ∎

LEMMA 19. *Let $K = A^T D^{-1} A$ correspond to a connected DC (real weights) network containing wires of two magnitudes—high admittance (functioning) wires and low admittance (nearly open circuit) wires. Let the preconditioner M be as above. Assume that the admittance of a typical functioning edge is ρ_1, and the admittance of a typical faulted edge is ρ_2. Then $\sigma(\tilde{K}, T)$ has an upper bound that is independent of ρ_1 and ρ_2.*

Proof. In this lemma, we consider the support of T for $\tilde{K}$, $\sigma(\tilde{K}, T)$. We map the edges of K (which are the same as the edges of $\tilde{K}$) onto paths in T using the same mapping as in Lemma 14. As in the previous case, the dilation is given by the longest path in T onto which an edge of K is mapped. This path can be as long as the longest path in T but is nevertheless independent of ρ_1 and ρ_2. For the congestion, let e be an edge in T and let $d_1, \ldots, d_k$ be the edges in K whose embedding path in T includes edge e. If e is a faulted edge, that is, $\text{weight}(e) \approx \rho_2$, then the d_i are faulted edges of K since only faulted edges of K would be mapped to paths that pass through the faulted edges of T. Therefore, $\sum_{i=1}^{k} \text{weight}(d_i) \approx k\rho_2$. Since the weight of e is also approximately ρ_2 and the congestion is defined as the previous sum divided by the weight of e, the congestion is independent of ρ_1 and ρ_2. If e is not a faulted edge, then some of the d_i may be faulted edges and some may not. In this case, $\sum_{i=1}^{k} \text{weight}(d_i) = O(\rho_1 + \rho_2) = O(\rho_1)$. Since e is not a faulted edge, its weight is approximately ρ_1, and the congestion is again independent of ρ_1 and ρ_2. Since the congestion and dilation separately are independent of ρ_1 and ρ_2, so is their product, which is an upper bound on the support of T for $\tilde{K}$, $\sigma(\tilde{K}, T)$. ∎

THEOREM 20. *Let $K = A^T D^{-1} A$ correspond to a connected DC (real weights) network containing wires of two magnitudes—high admittance (functioning) wires and low admittance (nearly open circuit) wires. Let the preconditioner M be as above. Assume that the admittance of a typical functioning edge is ρ_1, and the admittance of a typical faulted edge is ρ_2. Then $\kappa(M, K)$ has an upper bound that is independent of ρ_1 and ρ_2.*

Proof. From Lemma 15, we have $\kappa(M, K) \leq \sigma(\tilde{K}, T)\sigma(M, K)$, where K, $\tilde{K}$, T, and M are as before. We have shown in Lemmas 18 and 19 that the upper bounds on $\sigma(K, M)$ and $\sigma(\tilde{K}, T)$ are each independent of ρ_1 and ρ_2. Therefore, their product, which is an upper bound on the condition number of the preconditioned system, $\kappa(M, K)$, is also independent of ρ_1 and ρ_2. ∎

2.3.3. Extensions to analysis of AC networks. Finally, we extend the analysis to the AC network case.

DEFINITION 21. *Let B be a matrix of the form $B = A^T D^{-1} A$, where A is real with full column rank and D is a diagonal matrix. We say that B is* cone positive definite *if the diagonal elements of D lie in a pointed cone in the complex plane. In other words, if d_j is a diagonal element of D, then $d_j = x_j + iy_j$, where $x_j > 0$ and $|y_j| \leq \mu x_j$ for some positive cone constant μ.*

We assume that the original system $K = A^T D^{-1} A$ is cone positive definite. Note that satisfying this assumption simply requires that all of the impedances have some resistive component, which is true of any real power network. In this case, we can bound the condition number of the AC case by the condition number of the real part

of the system and a function of the cone constant. Strictly speaking, our matrices are not cone positive definite because of the 1-dimensional nullspace arising because the nodes are ungrounded. Again, this detail is not significant since our system matrix and our preconditioner have the same nullspace.

LEMMA 22. *If the original system matrix K is cone positive definite, so is the preconditioner M.*

Proof. The edge weights of the support tree are by construction sums of edge weights of the original graph. Therefore, the weighted Laplacian matrix T of the support tree is cone positive definite with the same cone constant as K. The preconditioner M is the Schur complement of T obtained by eliminating the internal tree nodes. Gaussian elimination performed on the internal nodes of the support tree is equivalent to performing series circuit reduction on those nodes [9]. If we reduce a node with edges connecting two nodes in series with admittances c_1 and c_2, the resulting edge has admittance $c_{new} = c_1c_2/(c_1 + c_2)$. Thus, if c_1 and c_2 are in the pointed cone with cone constant μ, so is c_{new} since

$$c_{new} = \frac{c_1c_2}{c_1 + c_2} = \frac{1}{\frac{1}{c_1} + \frac{1}{c_2}}, \tag{12}$$

and pointed cones are closed under addition and under taking reciprocals. Therefore, after eliminating all of the internal nodes of the support tree, we have a graph whose edge weights lie in the original pointed cone. The preconditioner M is the weighted Laplacian matrix of this graph; therefore, M is cone positive definite with the same cone constant μ as the original system matrix K. ∎

THEOREM 23. *In the AC case with impedances lying in a pointed cone in the complex plane, $\kappa(M, K) \leq (1 + \mu^2)\kappa(\mathrm{Re}(M), \mathrm{Re}(K))$, where μ is the cone constant.*

Proof. Assume that $K = A^TD^{-1}A$ as before and that $D^{-1} = E + iF$. Since the preconditioner M is also a weighted Laplacian matrix, we can write $M = R^T\Delta R$ where R is an NAI matrix and Δ is a diagonal matrix. As with the components of K, R is a real matrix made up of 1's, -1's, and 0's, and Δ is a diagonal matrix with complex diagonal entries lying in the same cone as the diagonal entries of D. Let $\Delta = B + iC$.

We first establish bounds on $|\mathbf{x}^*A^TD^{-1}A\mathbf{x}|$ in terms of the real part of D^{-1} and the cone constant μ. Let $\mathbf{w} = A\mathbf{x}$. Let d_j represent the diagonal elements of D^{-1}, e_j represent the diagonal elements of E, and f_j represent the diagonal elements of F. Then we have

$$\begin{aligned}
|\mathbf{x}^*A^TD^{-1}A\mathbf{x}| &= |\mathbf{w}^*D^{-1}\mathbf{w}| \\
&= \left|\sum_j |w_j|^2 \cdot d_j\right| \\
&\leq \sum_j \left(|w_j|^2 \cdot |d_j|\right) \\
&\leq \sum_j \left(|w_j|^2\sqrt{1+\mu^2} \cdot e_j\right) \\
&= |\mathbf{w}^*E\mathbf{w}|\sqrt{1+\mu^2} \\
&= |\mathbf{x}^*A^TEA\mathbf{x}|\sqrt{1+\mu^2}.
\end{aligned} \tag{13}$$

Next we bound $|\mathbf{x}^* A^T D^{-1} A\mathbf{x}|$ from below. In this case we have

$$\begin{aligned} |\mathbf{x}^* A^T D^{-1} A\mathbf{x}| &= |\mathbf{w}^* D^{-1}\mathbf{w}| \\ &= |\mathbf{w}^*(E+iF)\mathbf{w}| \\ &= \left|\sum |w_j|^2 e_j + i\sum |w_j|^2 f_j\right| \\ &\geq \left|\sum |w_j|^2 e_j\right| \\ &= |\mathbf{x}^* A^T E A\mathbf{x}|. \end{aligned} \tag{14}$$

The preconditioner M is also a weighted Laplacian matrix of the form $M = R^T \Delta R$ where R is an NAI matrix and Δ is a diagonal matrix. In addition, by Lemma 22, M is cone positive definite with the same cone constant μ as K. Therefore, the same bounds apply to M. Namely,

$$|\mathbf{x}^* R^T \Delta R\mathbf{x}| \leq |\mathbf{x}^* R^T C R\mathbf{x}|\sqrt{1+\mu^2}, \tag{15}$$

and

$$|\mathbf{x}^* R^T \Delta R\mathbf{x}| \geq |\mathbf{x}^* R^T C R\mathbf{x}|. \tag{16}$$

Using these bounds, we can establish an upper bound on the condition number of the preconditioned AC network in terms of the condition number of the real part of the network.

$$\begin{aligned} \kappa(M,K) &= \max \frac{|\mathbf{x}^* M\mathbf{x}|}{|\mathbf{x}^* K\mathbf{x}|} \cdot \max \frac{|\mathbf{x}^* K\mathbf{x}|}{|\mathbf{x}^* M\mathbf{x}|} \\ &\leq \max \frac{\sqrt{1+\mu^2}|\mathbf{x}^* R^T C R\mathbf{x}|}{|\mathbf{x}^* A^T E A\mathbf{x}|} \cdot \max \frac{\sqrt{1+\mu^2}|\mathbf{x}^* A^T E A\mathbf{x}|}{|\mathbf{x}^* R^T C R\mathbf{x}|} \\ &= (1+\mu^2) \max \frac{|\mathbf{x}^* R^T C R\mathbf{x}|}{|\mathbf{x}^* A^T E A\mathbf{x}|} \cdot \max \frac{|\mathbf{x}^* A^T E A\mathbf{x}|}{|\mathbf{x}^* R^T C R\mathbf{x}|} \\ &= (1+\mu^2)\kappa(\mathrm{Re}(M), \mathrm{Re}(K)). \qquad \square \end{aligned} \tag{17}$$

This analysis seemingly implies that we may as well precondition the AC network based only on its DC components. But our experiments indicate that keeping the imaginary part in the preconditioner gives better results. Therefore, there may exist a stronger analysis of the complex case.

3. Splitting. Since we are assuming that the gap between the magnitudes of high impedance wires and low impedance wires is large, and that removal of the high impedance wires disconnects the graph, the matrix $K = A^T D^{-1} A$ can be arbitrarily ill conditioned. Even though the preconditioner effectively reduces condition number, it does not lead to accurate solution by itself under these conditions. Again letting M be the preconditioner for K, the difficulty is that although the product $M^{-1}K$ is well conditioned, nonetheless the resulting vector $M^{-1}(K\mathbf{v})$ may be computed inaccurately because M^{-1} and K are very ill conditioned separately.

We solve this problem by splitting K into its "large space," i.e., the range of K_1 (where K_1 denotes, as before, the weighted Laplacian of the functioning wires) and its "small space," which is null(K_1). These spaces ("large" versus "small") are reversed for M^{-1}. We can easily identify these spaces from the graph and then split

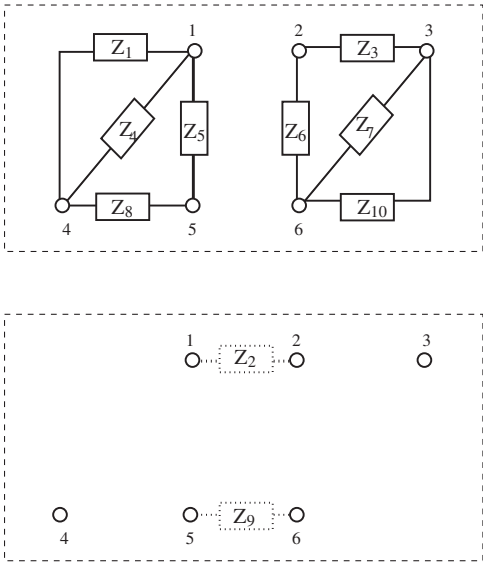

FIG. 7. *The same faulted AC network as shown in Figure* 2 *split into its functioning* (K_1) *and faulted* (K_2) *subgraphs. The top graph shows the functioning subgraphs, and the bottom graph shows the faulted subgraphs.*

$\mathbf{r} = (K_1+K_2)\mathbf{v}$ (in $O(n)$ time) into $\mathbf{r} = \mathbf{r}_R+\mathbf{r}_N$. We then compute $\mathbf{y} = M^{-1}(\mathbf{r}_R+\mathbf{r}_N)$ term by term, changing interim quantities that would be zero in exact arithmetic to zero, then recombining.

The matrix K_1 is just the weighted Laplacian of the functioning part of the system, and K_2 is the weighted Laplacian of the faulted part of the system. The original matrix K then equals $K_1 + K_2$. In Figure 7, we show these two graphs for the same network as shown in Figure 2.

For this network, $K_1 = A_1^T D_1^{-1} A_1$ and $K_2 = A_2^T D_2^{-1} A_2$, where

$$A_1 = \begin{bmatrix} 1 & & & -1 & & & \\ & 1 & -1 & & & & \\ 1 & & & -1 & & & \\ 1 & & & & -1 & & \\ & 1 & & & & -1 & \\ & & 1 & & & -1 & \\ & & & 1 & -1 & & \\ & & 1 & & & -1 & \end{bmatrix},$$

$$D_1 = \operatorname{diag}(\begin{bmatrix} Z_1 & Z_3 & Z_4 & Z_5 & Z_6 & Z_7 & Z_8 & Z_{10} \end{bmatrix}),$$

$$A_2 = \begin{bmatrix} 1 & -1 & & & \\ & & & 1 & -1 \end{bmatrix},$$

$$D_2 = \operatorname{diag}(\begin{bmatrix} Z_2 & Z_9 \end{bmatrix}). \tag{18}$$

In particular, our algorithm is as follows. Each time we form $M^{-1}(K\mathbf{v}) = M^{-1}(K_1 + K_2)\mathbf{v}$, we first split $(K_1 + K_2)\mathbf{v}$ into its subgraphs in range(K_1) and null(K_1). To do this, we form an orthogonal projector PP^T onto the null space of K_1. The projector PP^T is formed based on the graph structure of the network and can be applied in $O(n)$ time as follows. Removing the high impedance (faulty) wires

disconnects the graph into separate subgraphs of functioning wires, which we refer to as K_1-subgraphs. A vector in range(K_1) is defined by the property that its entries over each K_1-subgraph sum to zero. On the other hand, any vector in null(K_1) has the defining property that entries corresponding to nodes within a K_1-subgraph of the graph have the same value. Thus, we build an orthogonal projector onto null(K_1) using a connected subgraph search. Consider forming the product $\mathbf{r} = K\mathbf{v}$. We can split $\mathbf{r}$ according to: $\mathbf{r}_N = PP^TK\mathbf{v}$ and $\mathbf{r}_R = (I - PP^T)K\mathbf{v}$. We compute $\mathbf{r}_N$ as $PP^TK_2\mathbf{v}$ since the other term drops out. Similarly, we compute $\mathbf{r}_R$ as $K_1\mathbf{v} + K_2\mathbf{v} - PP^TK_2\mathbf{v}$. We then invoke our preconditioning algorithm with this split $\mathbf{r}$.

In invoking our preconditioning algorithm, as described in section 2, instead of solving $M\mathbf{y} = \mathbf{r}$, we solve the equivalent (but sparse) problem

$$LU \begin{bmatrix} \mathbf{y} \\ \mathbf{c} \end{bmatrix} = G\mathbf{r}, \tag{19}$$

where LU is the (sparse) LU factorization of T, and G is the matrix that extends a vector with zeros to be the size of the range space of T, i.e.,

$$G\mathbf{r} = \begin{bmatrix} \mathbf{r} \\ \mathbf{0} \end{bmatrix}. \tag{20}$$

We next forward solve for $\mathbf{u}_R = L^{-1}G\mathbf{r}_R$ and $\mathbf{u}_N = L^{-1}G\mathbf{r}_N$. Let S be the set of row numbers of nodes whose paths to the root contain only high-impedance edges. (Because of our definitions, S is exactly the root and children of the root.) Let J be the matrix that zeros out the components in S (i.e., J is a diagonal matrix with zeros in the diagonal positions corresponding to elements of S, and ones elsewhere on the main diagonal). We claim below that in exact arithmetic $\mathbf{u}_R = J\mathbf{u}_R$. Next we do the two backward solves, $\mathbf{w}_N = U^{-1}\mathbf{u}_N$ and $\mathbf{w}_R = U^{-1}J\mathbf{u}_R$. Note the important step in our algorithm of explicitly applying J to $\mathbf{u}_R$ before performing the back substitution.

Finally, we extract and return the part of $\mathbf{w}_R + \mathbf{w}_N$ that corresponds to the $\mathbf{y}$ in $M\mathbf{y} = \mathbf{v}$; i.e., $\mathbf{y} = G^T(\mathbf{w}_R + \mathbf{w}_N)$.

Thus, our algorithm for evaluating $\mathbf{y} = M^{-1}K\mathbf{v}$ is summarized as follows:

$$\begin{aligned} \mathbf{r}_N &= PP^TK_2\mathbf{v}, \\ \mathbf{r}_R &= K_1\mathbf{v} + K_2\mathbf{v} - PP^TK_2\mathbf{v}, \\ \mathbf{u}_N &= L^{-1}G\mathbf{r}_N, \\ \mathbf{u}_R &= L^{-1}G\mathbf{r}_R, \\ \mathbf{w}_N &= U^{-1}\mathbf{u}_N, \\ \mathbf{w}_R &= U^{-1}J\mathbf{u}_R, \\ \mathbf{y} &= G^T(\mathbf{w}_R + \mathbf{w}_N). \end{aligned} \tag{21}$$

Let us now provide three preliminary lemmas about this construction.

LEMMA 24. *Let L and U be lower and upper triangular matrices such that $T = LU$, and let Δ be a diagonal matrix consisting of the main diagonal elements of U. Let $\tilde{U} = \Delta^{-1}U$. Applying L^{-1} sums the entries from the leaves to the root of the support tree. The matrices $\tilde{U}$ and L have the properties that $\|\tilde{U}^{-1}\| \le c_n$, $\|\tilde{U}\| \le d_n$, $\|L^{-1}\| \le k_n$, and $\|L\| \le f_n$, where c_n, d_n, f_n, and k_n are constants that depend only on n.*

Proof. These properties can all be proven combinatorially. We omit the proof. ∎

LEMMA 25. *In exact arithmetic, $J\mathbf{u}_R = \mathbf{u}_R$, where $\mathbf{u}_R$ is defined in* (21).

Proof. Let n_i be a node in the support tree, and let N_i be the set of leaves rooted at n_i. Recall that the leaves of the support tree correspond directly to the nodes of the original network. The weight of the parent edge of n_i is constructed (as in Gremban) to be the sum of the edges connecting the original graph nodes N_i to the rest of the graph. This sum is known as the *frontier* of N_i. Since in our algorithm we force all high impedance edges to be in the top level graph separator, the nodes in the support tree indexed by elements of S are those whose corresponding nodes in the original graph are connected to the rest of the graph by only high impedance edges. This implies that leaves rooted at such a node make up one or more K_1-subgraphs.

Since applying L^{-1} sums the entries from the leaves to the root of the support tree, if $\mathbf{u}_R = L^{-1}G\mathbf{r}_R$, for example, then $u_R(i)$ equals the sum of the entries in $\mathbf{r}_R$ associated with leaves rooted at n_i. As mentioned above, entries in $\mathbf{r}_R$ corresponding to a K_1-subgraph sum to zero since $\mathbf{r}_R \in \text{range}(K_1)$. Therefore, if $i \in S$, then $u_R(i) = 0$ in exact arithmetic. ∎

Let the constants α and β be defined as

$$\alpha = \max\left(\frac{\max_i(D_1^{-1}(i,i))}{\min_i(D_1^{-1}(i,i))}, \frac{\max_i(D_2^{-1}(i,i))}{\min_i(D_2^{-1}(i,i))}\right), \qquad (22)$$
$$\beta = \max_i(D_1^{-1}(i,i)),$$

where the $D_1^{-1}(i,i)$ are the weights of the low impedance edges, and the $D_2^{-1}(i,i)$ are the weights of the high impedance edges.

LEMMA 26. *$\|\Delta^{-1}J\| \le c_n \cdot \alpha/\beta$, where c_n is a constant depending only on n.*

Proof. The elements of Δ are either sums of low impedance weights, sums of high impedance weights, or a mixture of low and high impedance weights. The elements of Δ consisting of the sums of high impedance weights correspond to the row numbers of nodes whose paths to the root contain only high impedance edges, i.e., elements whose associated entries in $\mathbf{r}_R$ correspond to a K_1-subgraph. As shown in Lemma 25, these are the same elements of J that are set to zero. Therefore, the product $\Delta^{-1}J$ is diagonal with the only nonzero elements being sums of low impedance weights. Therefore, $\|\Delta^{-1}J\|$ is equal to the reciprocal of a sum of low impedance (high admittance) weights. The number of weights in the sum is bounded by the number of nodes in the support tree. Therefore, $\|\Delta^{-1}J\| \le c_n \cdot 1/\min_i(D_1^{-1}(i,i)) \le c_n \cdot \alpha/\beta$. ∎

Now we can state our main theorem for this section.

THEOREM 27. *Using the algorithm defined by* (21) *in the presence of roundoff error, the computed $M^{-1}(K\mathbf{v})$ has the form $(M^{-1}K + E)\mathbf{v}$, where E satisfies $\|E\| \le \epsilon_{mach}c_nc_m\alpha + O({\epsilon_{mach}}^2)$, and ϵ_{mach} is machine epsilon.*

Remark 3. In contrast, if we compute $M^{-1}K\mathbf{v}$ in the naive manner, the computed result has the form $(M^{-1}K + E)\mathbf{v}$, where $\|E\| \le \|K\| \cdot \|M^{-1}\| \cdot \epsilon_{mach}$, which could be very large.

Proof. The heart of this proof is in the following two lemmas, which show that $\mathbf{w}_R$ and $\mathbf{w}_N$ are both computed accurately.

LEMMA 28. *The computed $\mathbf{w}_R$ has the form $\hat{\mathbf{w}}_R = [(U^{-1}JL^{-1}G(I - PP^T)(K_1 + K_2)) + E_R]\mathbf{v}$, where $\|E_R\| \le \epsilon_{mach} \cdot c_n \cdot \alpha/\beta \cdot \|K_1 + K_2\| + O({\epsilon_{mach}}^2)$ and α and β are defined by* (22).

Proof. Note that we will reuse the symbol c_n in this proof to mean a constant depending only on n whose value may change from statement to statement.

The range space component of $\mathbf{r}$ is given by $\mathbf{r}_R = K_1\mathbf{v} + K_2\mathbf{v} - PP^TK_2\mathbf{v} = (I - PP^T)(K_1+K_2)\mathbf{v}$, and the computed $\mathbf{r}_R$ has the form $\hat{\mathbf{r}}_R = [(I-PP^T)(K_1+K_2)+E']\mathbf{v}$, where $\|E'\| \le \|K_1\| \cdot \epsilon_{mach} \cdot c_n$. We are assuming here that $\|K_2\| \ll \|K_1\|$.

We next extend $\mathbf{r}_R$ with zeros (i.e., apply G), and perform the forward solve $\mathbf{u}_R = L^{-1}G\mathbf{r}_R$. Note that $\mathbf{u}_R$ is no longer necessarily in range(K_1). The computed results again have the form $\hat{\mathbf{u}}_R = (L^{-1}G + E''')\hat{\mathbf{r}}_R$, where $\|E'''\| \leq \|L^{-1}\| \cdot \epsilon_{mach} \cdot c_n$.

Next, let J be the matrix that sets to zero the elements of $\hat{\mathbf{u}}_R$ that would have been zero in exact arithmetic. (This step is explained in Lemma 25.)

Now we do the back solve $\mathbf{w}_R = U^{-1}J\mathbf{u}_R$. The computed $\mathbf{w}_R$ has the form $(U + F)\hat{\mathbf{w}}_R = J\hat{\mathbf{u}}_R$, where $|F| \leq |U| \cdot \epsilon_{mach} \cdot c_n$ entrywise. Rearranging we get $\hat{\mathbf{w}}_R = (U + F)^{-1}J\hat{\mathbf{u}}_R$. Substituting the Taylor series approximation for $(U + F)^{-1}$ and dropping the high-order terms, we have $\hat{\mathbf{w}}_R = (U^{-1} + U^{-1}FU^{-1})J\hat{\mathbf{u}}_R$.

Recall that $U = \Delta\tilde{U}$, where Δ is a diagonal matrix and $\tilde{U}$ is well conditioned. Since $|F| \leq |U| \cdot \epsilon_{mach} \cdot c_n$ entrywise, F has the same structure, i.e., $F = \Delta\tilde{F}$, where Δ is the same diagonal matrix and $|\tilde{F}| \leq |\tilde{U}| \cdot \epsilon_{mach} \cdot c_n$ entrywise. We can now write the computed $\mathbf{w}_R$ as $\hat{\mathbf{w}}_R = (\tilde{U}^{-1} + \tilde{U}^{-1}\tilde{F}\tilde{U}^{-1})\Delta^{-1}J\hat{\mathbf{u}}_R$.

Putting all of the steps together, we have that

$$\begin{aligned}\hat{\mathbf{w}}_R &= (\tilde{U}^{-1} + \tilde{U}^{-1}\tilde{F}\tilde{U}^{-1})(\Delta^{-1}J)(L^{-1}G + E''') \\ &\qquad \cdot [(I - PP^T)(K_1 + K_2) + E']\mathbf{v} \\ &= [(U^{-1}JL^{-1}G(I - PP^T)(K_1 + K_2)) + E_R]\mathbf{v},\end{aligned} \tag{23}$$

where

$$\begin{aligned}E_R = {} & \tilde{U}^{-1}\tilde{F}\tilde{U}^{-1}\Delta^{-1}JL^{-1}G(I - PP^T)(K_1 + K_2) \\ & + \tilde{U}^{-1}\Delta^{-1}JL^{-1}GE' \\ & + \tilde{U}^{-1}\Delta^{-1}JE'''(I - PP^T)(K_1 + K_2) \\ & + O(\epsilon_{mach}{}^2).\end{aligned} \tag{24}$$

Thus we have

$$\begin{aligned}\|E_R\| &\leq \|\tilde{U}^{-1}\tilde{F}\tilde{U}^{-1}\Delta^{-1}JL^{-1}G(I - PP^T)(K_1 + K_2)\| \\ &\quad + \|\tilde{U}^{-1}\Delta^{-1}JL^{-1}GE'\| \\ &\quad + \|\tilde{U}^{-1}\Delta^{-1}JE'''(I - PP^T)(K_1 + K_2)\| \\ &\quad + O(\epsilon_{mach}{}^2) \\ &\leq \|\tilde{U}^{-1}\tilde{F}\tilde{U}^{-1}\| \cdot \|\Delta^{-1}J\| \cdot \|L^{-1}G(I - PP^T)(K_1 + K_2)\| \\ &\quad + \|\tilde{U}^{-1}\| \cdot \|\Delta^{-1}J\| \cdot \|L^{-1}G\| \cdot \|E'\| \\ &\quad + \|\tilde{U}^{-1}\| \cdot \|\Delta^{-1}J\| \cdot \|E'''\| \cdot \|(I - PP^T)(K_1 + K_2)\| \\ &\quad + O(\epsilon_{mach}{}^2) \\ &\leq \epsilon_{mach} \cdot c_n \cdot \alpha/\beta \cdot \|K_1 + K_2\| + O(\epsilon_{mach}{}^2),\end{aligned} \tag{25}$$

where α and β are as described for Lemma 26. ∎

LEMMA 29. *The computed* $\mathbf{w}_N$ *has the form* $\hat{\mathbf{w}}_N = [(U^{-1}L^{-1}GPP^TK_2) + E_N]\mathbf{v}$, *where* $\|E_N\| \leq \epsilon_{mach}c_n\frac{\alpha}{\beta}\|K_1\| + O(\epsilon_{mach}{}^2)$ *and* α *and* β *are defined by* (22).

Proof. The computed $\mathbf{r}_N$ has the form $\hat{\mathbf{r}}_N = (PP^TK_2 + E^0)\mathbf{v}$, where $\|E^0\| \leq \|K_2\| \cdot \epsilon_{mach} \cdot c_n$, and c_n is a small constant that depends only on n. This follows from the standard properties of matrix-vector multiplication and the fact that $\|PP^T\| = 1$. Note that we will reuse the symbol c_n in this proof to mean a constant depending only on n whose value may change from statement to statement.

We next extend $\mathbf{r}_N$ with zeros (i.e., apply G), and perform the forward solve $\mathbf{u}_N = L^{-1}G\mathbf{r}_N$. Note that $\mathbf{u}_N$ is no longer necessarily in null(K_1).

The computed results again have the form $\hat{\mathbf{u}}_N = (L^{-1}G + E'')\hat{\mathbf{r}}_N$, where $\|E''\| \leq \|L^{-1}\| \cdot \epsilon_{mach} \cdot c_n$.

Now we do the back solve, $\mathbf{w}_N = U^{-1}\mathbf{u}_N$. Similarly to the case for $\mathbf{w}_R$, for $\mathbf{w}_N$ we have

$$\begin{aligned}\hat{\mathbf{w}}_N &= (U^{-1} + U^{-1}FU^{-1})(L^{-1}G + E'')\hat{\mathbf{r}}_N \\ &= (U^{-1} + U^{-1}FU^{-1})(L^{-1}G + E'')(PP^TK_2 + E^0)\mathbf{v} \\ &= (\tilde{U}^{-1} + \tilde{U}^{-1}\tilde{F}\tilde{U}^{-1})\Delta^{-1}(L^{-1}G + E'')(PP^TK_2 + E^0)\mathbf{v} \\ &= [(U^{-1}L^{-1}GPP^TK_2) + E_N]\mathbf{v},\end{aligned} \tag{26}$$

where

$$\begin{aligned}E_N = \tilde{U}^{-1}\Delta^{-1}E''PP^TK_2 &+ \tilde{U}^{-1}\Delta^{-1}L^{-1}GE^0 \\ &+ \tilde{U}^{-1}\tilde{F}\tilde{U}^{-1}\Delta^{-1}L^{-1}GPP^TK_2 \\ &+ O(\epsilon_{mach}{}^2).\end{aligned} \tag{27}$$

So we have

$$\begin{aligned}\|E_N\| \leq \|\tilde{U}^{-1}\| \cdot \|\Delta^{-1}E''PP^TK_2\| &+ \|\tilde{U}^{-1}\| \cdot \|\Delta^{-1}L^{-1}GE^0\| \\ &+ \|\tilde{U}^{-1}\| \cdot \|\tilde{F}\| \cdot \|\tilde{U}^{-1}\| \cdot \|\Delta^{-1}L^{-1}GPP^TK_2\| \\ &+ O(\epsilon_{mach}{}^2).\end{aligned} \tag{28}$$

We have shown in Lemma 24 that $\|\tilde{U}^{-1}\| \leq c_n$ and $\|L^{-1}\| \leq c_n$. As above, $\|E''\| \leq \epsilon_{mach} \cdot c_n\|L^{-1}\|$ and $\|E^0\| \leq \epsilon_{mach} \cdot c_n \cdot \|K_2\|$. Finally, we are assuming that the high impedance weights are significantly larger than the low impedance weights, i.e., that $\|K_2\| \leq \rho\|K_1\|$ for a small constant ρ.

Since Δ is a diagonal matrix with sums of high impedance and low impedance weights on its diagonal, $\|\Delta^{-1}\| \leq 1/\rho \cdot \max_i(1/D_1^{-1}(i,i))$, where $D_1^{-1}(i,i)$ are the weights of the low impedance edges. Therefore, $\|\Delta^{-1}\| \leq \alpha/(\rho\beta)$, where α and β are defined by (22).

Therefore, we can bound the error term $\|E_N\|$ as

$$\begin{aligned}\|E_N\| &\leq c_n\frac{1}{\rho}\frac{\alpha}{\beta}\epsilon_{mach}\rho\|K_1\| + O(\epsilon_{mach}{}^2) \\ &\leq \epsilon_{mach}c_n\frac{\alpha}{\beta}\|K_1\| + O(\epsilon_{mach}{}^2). \qquad \square\end{aligned} \tag{29}$$

Now finally, we conclude the proof of the main theorem. We have shown in Lemmas 28 and 29 that the computed $\mathbf{w}_R$ has the form $\hat{\mathbf{w}}_R = [(U^{-1}JL^{-1}G(I - PP^T)(K_1 + K_2)) + E_R]\mathbf{v}$, where $\|E_R\| \leq \epsilon_{mach} \cdot c_n \cdot \alpha/\beta \cdot \|K_1 + K_2\| + O(\epsilon_{mach}{}^2)$, and the computed $\mathbf{w}_N$ has the form $\hat{\mathbf{w}}_N = [(U^{-1}L^{-1}GPP^TK_2) + E_N]\mathbf{v}$, where $\|E_N\| \leq$

$\epsilon_{mach} c_n \frac{\alpha}{\beta} \|K_1\| + O(\epsilon_{mach}{}^2)$. Therefore, the total computed $\mathbf{w}$ has the form

$$\begin{aligned} \hat{\mathbf{w}} &= \hat{\mathbf{w}}_R + \hat{\mathbf{w}}_N \\ &= [(U^{-1}JL^{-1}G(I - PP^T)(K_1 + K_2)) + E_R]\mathbf{v} \\ &\quad + [(U^{-1}L^{-1}GPP^T(K_1 + K_2)) + E_N]\mathbf{v} \\ &= (M^{-1}K + E)\mathbf{v}, \end{aligned} \tag{30}$$

where

$$\begin{aligned} \|E\| &= \|E_R + E_N\| \\ &\leq \epsilon_{mach} c_n \frac{\alpha}{\beta} \|K\| + O(\epsilon_{mach}{}^2). \end{aligned} \tag{31}$$

Finally, since $\beta = \max_i(D_1^{-1}(i,i))$ and the functioning admittances are assumed to be larger than the faulty admittances, β is the maximum of all of the admittances. In particular, $\|D^{-1}\|/\beta = 1$; therefore, $\|K\|/\beta = c_m$, where c_m is a constant depending on the number of edges in the graph. Therefore, we have

$$\|E\| \leq \epsilon_{mach} c_n c_m \alpha + O(\epsilon_{mach}{}^2). \qquad \square \tag{32}$$

4. Computational experiments. We have tested this algorithm on two qualitatively different graphs: grid-graphs and sample power network graphs from the *MATPOWER* [21] power flow simulation package.

In our analysis, we assume that all of the high-impedance (nonfunctioning) edges are included in the top level of the graph separator. For these examples we defined the high-impedance edges to be exactly those edges cut in the top level graph partition. All remaining edges were taken to be low-impedance (functioning) edges. The low-impedance weights were chosen uniformly at random from a disk in $\mathbf{C}$ centered at 2 of radius 1. The high-impedance (low-admittance) weights were chosen uniformly at random from a disk centered at $2 \cdot 10^{-10}$ of radius $1 \cdot 10^{-10}$. For a 16,384-node system, this choice results in faulted systems of with condition number of approximately 10^{18} (as reported by MATLAB's *condest()* function). We tried several size problems of this kind, with similar results.

All experiments were conducted in MATLAB using sparse matrix operations. The extended preconditioner T was LU-factored without pivoting (and, as mentioned earlier, with no fill-in). The graph partitioning was done using the MATLAB Mesh Partitioning Toolbox [8].

Exact solutions were computed by first forming a random solution vector $\mathbf{x}$ and computing $\mathbf{b} = A\mathbf{x}$. We then used our splitting technique to compute $M^{-1}A^TD^{-1}\mathbf{b}$ accurately to initialize the iterative method.

The results we present in this section are with our algorithm implemented in TFQMR [6]. We have also had success with BiCG [19] and GMRES [16] (not reported here). Note that CG and LSQR are not applicable since (1) is not Hermitian. In addition, we do not use a method such as CSQMR [5] that can take advantage of the complex symmetry of the problem. We have special techniques for computing $M^{-1}(K\mathbf{v})$ accurately and cannot easily reuse our existing methods to compute $K^T(M^{-T}\mathbf{v})$. A transpose method could almost certainly be developed using similar ideas, but we have not tried to implement such a method.

We first compare our algorithm with other iterative methods. In all of the tests that follow, the stopping tolerance is $\|residual\|_2 < (10^{-10} * \|\mathbf{b}\|_2)$.

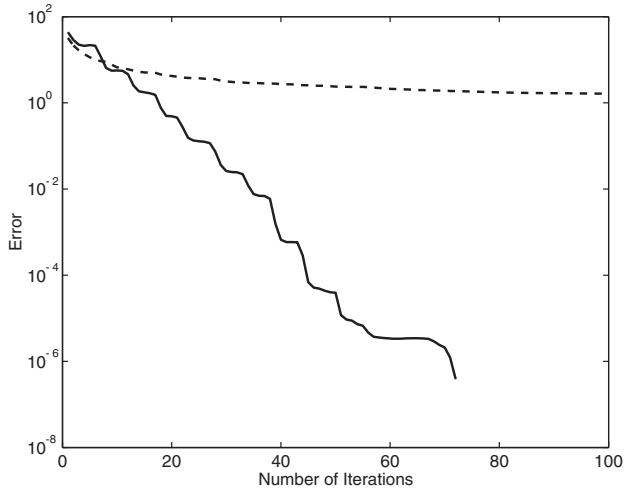

FIG. 8. *Our algorithm and unpreconditioned TFQMR applied to the same faulted* 16,384*-node AC network problem on a grid-graph. The dashed line is unpreconditioned TFQMR without splitting; the solid line is our preconditioned TFQMR with splitting.*

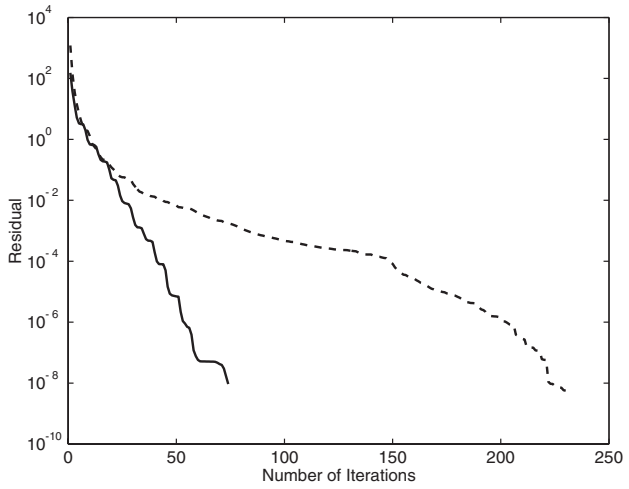

FIG. 9. *Residuals from our algorithm and unpreconditioned TFQMR applied to the same faulted* 16,384*-node AC network problem on a grid-graph. The dashed line is unpreconditioned TFQMR without splitting; the solid line is our preconditioned TFQMR with splitting.*

Figure 8 shows the results from a 16,384-node grid-graph and compares unpreconditioned TFQMR (without splitting) and preconditioned TFQMR with splitting. We see from the figure that our preconditioned TFQMR with splitting is far superior in terms of reducing the error. For TFQMR without preconditioning or splitting, the error was hardly reduced at all from the initial guess. It is important to note that we are plotting error $\|\mathbf{x}^{(j)} - \mathbf{x}\|$ rather than residual $\|K\mathbf{x}^{(j)} - \mathbf{b}\|$. The residual for such an ill-conditioned problem does not give a valid picture of convergence. We can see, for example, in Figure 9 that unpreconditioned TFQMR without splitting does in fact eventually converge (after approximately 250 iterations) in terms of reducing

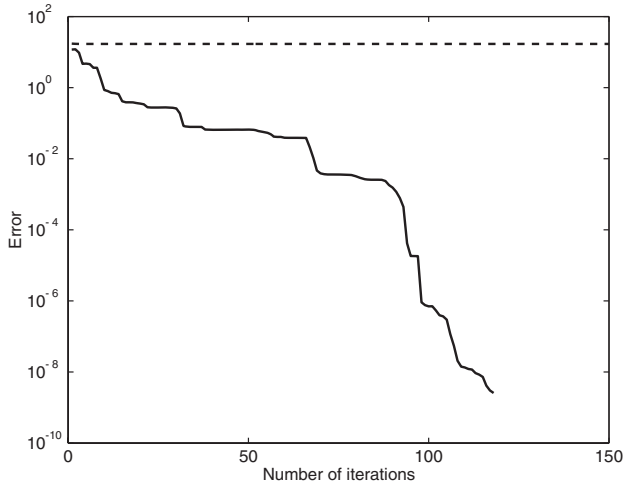

FIG. 10. *Our algorithm and unpreconditioned TFQMR applied to the same* 984*-node AC network problem on a graph from MATPOWER* [21]. *The dashed line is unpreconditioned TFQMR without splitting; the solid line is our preconditioned TFQMR with splitting.*

the updated residual to the specified tolerance, but the error stagnates at approximately 1.2 and never improves. In this example, TFQMR with our preconditioning and splitting took approximately 172.56 seconds, i.e., approximately 2.36 seconds per iteration. All remaining examples in this section will plot the error $\|\mathbf{x}^{(j)} - \mathbf{x}\|$.

Figure 10 shows the results from a 984-node network example from the *MATPOWER* [21] power flow simulation package. As in the previous example, our preconditioned TFQMR with splitting is far superior in terms of reducing the error. For TFQMR without preconditioning or splitting, the error was hardly reduced at all from the initial guess, stagnating at approximately 6.2. In this example, our algorithm took approximately 119.17 seconds (0.99 seconds per iteration) and reduced the error to approximately $2.58 \cdot 10^{-9}$.

Next we compare our algorithm to TFQMR with an incomplete LU (ILU) preconditioner. In Figure 11, we compare TFQMR with an ILU preconditioner (with no fill) to our algorithm on a 16,384-node AC network with no faults. In this example, our algorithm and TFQMR with ILU perform similarly. We use MATLAB's *luinc()* function for the incomplete LU factorization. In this example, unpreconditioned TFQMR without splitting took 180.28 seconds (0.41 seconds per iteration), TFQMR with ILU preconditioning took 75.15 seconds (0.62 seconds per iteration), and our algorithm took 206.84 seconds (2.35 seconds per iteration). In Figure 12, we make the same comparison on a system with faults. Again we see that in a faulted system, our algorithm is far superior in terms of reducing the error. As expected, in unpreconditioned TFQMR without splitting and TFQMR with ILU preconditioning the error stagnates fairly quickly and is never significantly reduced. Our algorithm took 172.56 seconds (2.36 seconds per iteration) and reduced the error substantially.

Next, we examine the effect of the splitting as separate from the support-tree preconditioning. In Figure 13 we compare the results of solving a faulted 16,384-node AC network problem on a grid-graph with and without splitting. This example shows the critical importance of the splitting technique. Although the support-tree preconditioner is a good preconditioner in the sense of improving the condition number

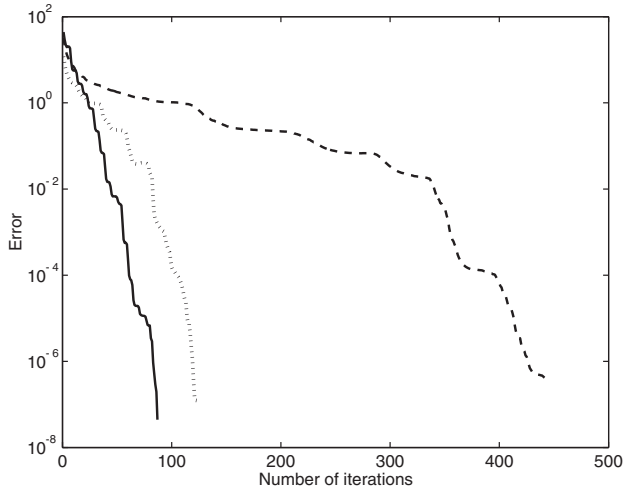

FIG. 11. *Our algorithm, unpreconditioned TFQMR, and TFQMR with ILU preconditioning applied to the same unfaulted* 16,384*-node AC network problem on a grid-graph. The dashed line is unpreconditioned TFQMR without splitting; the dotted line is TFQMR with an ILU preconditioner (no fill); and the solid line is our preconditioned TFQMR with splitting.*

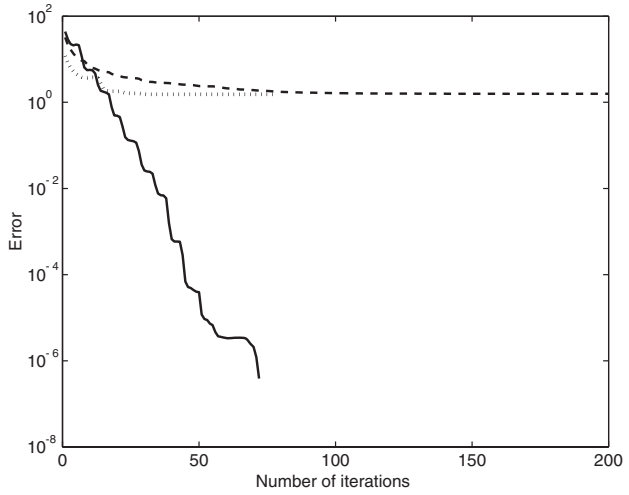

FIG. 12. *Our algorithm, unpreconditioned TFQMR, and TFQMR with ILU preconditioning applied to the same faulted* 16,384*-node AC network problem on a grid-graph. The dashed line is unpreconditioned TFQMR without splitting; the dotted line is TFQMR with an ILU preconditioner (no fill); and the solid line is our preconditioned TFQMR with splitting.*

of the system, it is not helpful unless it can be applied accurately.

Note that we do not perform the opposite experiment of applying our splitting technique without the support-tree preconditioner or with another preconditioner. Our splitting technique is tightly bound to the support-tree preconditioner, and it is not clear how to apply it to an arbitrary preconditioner.

The convergence rate of TFQMR is not known to be related to the condition number in Definition 3 (or indeed, to any condition number). Nonetheless, our experience

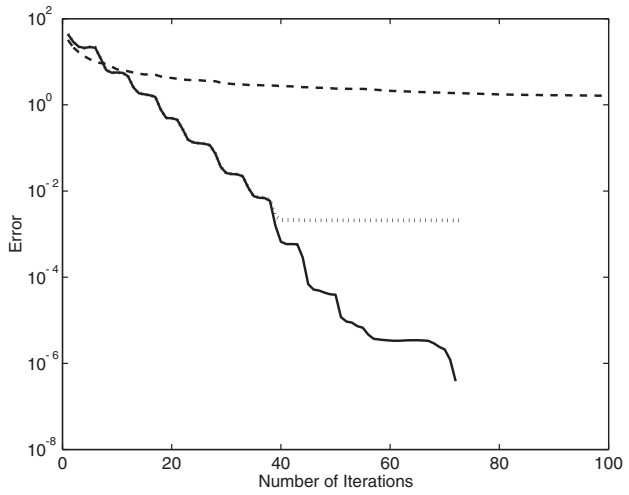

FIG. 13. *Our algorithm, unpreconditioned TFQMR, and TFQMR with our preconditioner but no splitting applied to the same faulted* 16,384*-node AC network problem on a grid-graph. The dashed line shows unpreconditioned TFQMR without splitting; the dotted line is TFQMR with support-tree preconditioning but without splitting; and the solid line is our preconditioned TFQMR with splitting.*

indicates that these two are related. Note that the condition number in Definition 3 is related to the condition number of $(M^*M)^{-1}(K^*K)$, which is the condition number that would apply if we were using CGNR.

Finally, we compare our algorithm with two direct methods, MATLAB's *backslash* command and the complete orthogonal decomposition (COD) method of Hough and Vavasis [13]. On the unfaulted 16,384-node grid-graph problem, MATLAB's *backslash* command is both fast and accurate, taking 9.21 seconds and resulting in a solution with error $2.96 \cdot 10^{-11}$. On the same unfaulted problem, our algorithm takes 203.76 seconds with a resulting error of $4.41 \cdot 10^{-8}$. On the faulted system, however, although *backslash* is still fast (9.23 seconds), the resulting solution has an error of $1.57 \cdot 10^{-3}$. Note that one step of iterative refinement improved this error by approximately one order of magnitude. However further steps of iterative refinement produced no further improvement in the error. In contrast, our algorithm takes 169.42 seconds with a resulting error of $3.87 \cdot 10^{-7}$ on the same faulted system.

The COD algorithm requires SPD systems, so we compare our algorithm with COD using real values for the impedances. Analogously to the complex-symmetric case, low-impedance weights were chosen uniformly at random from an interval on the real line centered at 2 of radius 1. High-impedance weights were chosen from an interval centered at $2 \cdot 10^{-10}$ of radius $1 \cdot 10^{-10}$. The COD algorithm is accurate, even in the presence of faults, but is significantly slower than our algorithm. On an unfaulted 256-node grid system with real edge weights, the COD algorithm took 117.38 seconds with a resulting error of $7.55 \cdot 10^{-15}$, MATLAB's *backslash* command took approximately 0.04 seconds with a resulting error of $2.6 \cdot 10^{-14}$, and our algorithm took approximately 0.94 seconds with a resulting error of $4.64 \cdot 10^{-9}$.

Once we introduce faults, our algorithm and the COD algorithm remain accurate, but the COD algorithm is still significantly slower. On a faulted 1024-node systems with real edge weights, the COD algorithm took 117.45 seconds with a resulting error of $4.94 \cdot 10^{-15}$, MATLAB's *backslash* command took approximately 0.014 seconds with

a resulting error of $5.96 \cdot 10^{-5}$, and our algorithm took approximately 0.84 seconds with a resulting error of $2.23 \cdot 10^{-8}$.

5. Conclusions. We have presented an iterative method for solving complex-symmetric linear systems arising in electrical power networks. We extend Gremban, Miller, and Zagha's [10] support-tree preconditioner to handle the case of faulted AC networks, i.e., complex weights and vastly different admittances. In addition to these extensions, we present a splitting technique that allows us to apply the preconditioner accurately even when the system matrix, and therefore the preconditioner, is arbitrarily ill conditioned. Our computational results show that this iterative method works well in practice in reducing the error.

Note that these results can also apply to the sandstone and shale problem of Vuik [20] and may also apply to interior point methods if we have nullspace information.

REFERENCES

[1] A. R. Bergen, *Power Systems Analysis*, Prentice–Hall, Englewood Cliffs, NJ, 1986.

[2] M. Bern, J. R. Gilbert, B. Hendrickson, N. Nguyen, and S. Toledo, *Support-graph preconditioners*, SIAM J. Matrix Anal. Appl., in review.

[3] E. Y. Bobrovnikova and S. A. Vavasis, *Accurate solution of weighted least squares by iterative methods*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1153–1174.

[4] A. J. Flueck and H.-D. Chiang, *Solving the nonlinear power flow equations with an inexact Newton method using GMRES*, IEEE Trans. Power Systems, 13 (1998), pp. 267–273.

[5] R. W. Freund, *Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 425–448.

[6] R. W. Freund, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470–482.

[7] F. D. Galiana, H. Javidi, and S. McFee, *On the application of a pre-conditioned conjugate gradient algorithm to power network analysis*, IEEE Trans. Power Systems, 9 (1994), pp. 629–636.

[8] J. R. Gilbert, G. L. Miller, and S.-H. Teng, *Geometric mesh partitioning: Implementation and experiments*, SIAM J. Sci. Comput., 19 (1998), pp. 2091–2110.

[9] K. D. Gremban, *Combinatorial Preconditioners for Sparse, Symmetric, Diagonally Dominant Linear Systems*, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, 1996.

[10] K. D. Gremban, G. L. Miller, and M. Zagha, *Performance evaluation of a new parallel preconditioner*, in Proceedings of the International Parallel Processing Symposium, IEEE Computer Society, Los Alamitos, CA, 1995, pp. 65–69.

[11] R. Guo and R. D. Skeel, *An algebraic hierarchical basis preconditioner*, Appl. Numer. Math., 9 (1992), pp. 21–32.

[12] E. V. Haynsworth, *Applications of an inequality for the Schur complement*, Proc. Amer. Math. Soc., 24 (1970), pp. 512–516.

[13] P. D. Hough and S. A. Vavasis, *Complete orthogonal decomposition for weighted least squares*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 369–392.

[14] V. E. Howle, *Efficient Iterative Methods for Ill-conditioned Linear and Nonlinear Network Problems*, Ph.D. thesis, Cornell University, Ithaca, NY, 2001.

[15] M. A. Pai, P. W. Sauer, and A. Y. Kulkarni, *A preconditioned iterative solver for dynamic simulation of power systems*, in Proceedings of the IEEE International Symposium on Circuits and Systems, IEEE, New York, 1995, pp. 1279–1282.

[16] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[17] A. Semlyen, *Fundamental concepts of a Krylov subspace power flow methodology*, IEEE Trans. Power Systems, 11 (1996), pp. 1528–1537.

[18] P. M. Vaidya, *Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners*, manuscript. A talk based on the manuscript was presented at the IMA Workshop on Graph Theory and Sparse Matrix Computation, Minneapolis, MN, 1991.

[19] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.
[20] C. Vuik, A. Segal, and J. A. Meijerink, *An efficient preconditioned CG method for the solution of a class of layered problems with extreme contrasts in the coefficients*, J. Comput. Phys., 152 (1999), pp. 385–403.
[21] R. Zimmerman and D. Gan, *MATPOWER*, http://www.pserc.cornell.edu/powerweb/.