AMPS: AN AUGMENTED MATRIX FORMULATION FOR PRINCIPAL SUBMATRIX UPDATES WITH APPLICATION TO POWER GRIDS*

YU-HONG YEUNG†, ALEX POTHEN†, MAHANTESH HALAPPANAVAR‡, AND ZHENYU HUANG‡

Abstract. We present AMPS, an augmented matrix approach to update the solution to a linear system of equations when the matrix is modified by a few elements within a principal submatrix. This problem arises in the dynamic security analysis of a power grid, where operators need to perform $N-k$ contingency analysis, i.e., determine the state of the system when exactly k links from N fail. Our algorithms augment the matrix to account for the changes in it, and then compute the solution to the augmented system without refactoring the modified matrix. We provide two algorithms, a direct method, and a hybrid direct-iterative method for solving the augmented system. We also exploit the sparsity of the matrices and vectors to accelerate the overall computation. We analyze the time complexity of both algorithms, and show that it is bounded by the number of nonzeros in a subset of the columns of the Cholesky factor that are selected by the nonzeros in the sparse right-hand-side vector. Our algorithms are compared on three power grids with PARDISO, a parallel direct solver, and CHOLMOD, a direct solver with the ability to modify the Cholesky factors of the matrix. We show that our augmented algorithms outperform PARDISO (by two orders of magnitude), and CHOLMOD (by a factor of up to 5). Further, our algorithms scale better than CHOLMOD as the number of elements updated increases. The solutions are computed with high accuracy. Our algorithms are capable of computing $N-k$ contingency analysis on a 778 thousand bus grid, updating a solution with $k = 20$ elements in 16 milliseconds on an Intel Xeon processor.

Key words. direct methods, iterative methods, augmented matrix, sparse matrices, matrix updates, powerflow analysis, contingency analysis

AMS subject classifications. 65F50, 65F10, 65F05, 65Y20

1. Introduction.

We consider updating the solution to a system of equations $A\boldsymbol{x} = \boldsymbol{b}$, where A is a symmetric positive definite or indefinite $n \times n$ matrix and $\boldsymbol{b}$ is an n-vector, when a low-rank change is made to A. The change we consider is an update of a principal submatrix of the form

$$\hat{A} = A - HEH^\top, \tag{1}$$

where E is a symmetric $m \times m$ matrix, and H is an $n \times m$ submatrix of an identity matrix, and $m \ll n$. Since both A and E are symmetric, $\hat{A}$ is also symmetric. Note that the dimension of the matrix does not change when it is updated. We wish to compute the solution to the updated system

$$\hat{A}\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}. \tag{2}$$

We describe an augmented matrix approach to the solution of the updated system, in which the augmented matrix is a block 3×3 matrix whose $(1,1)$-block is the original matrix A, and the updates to A are represented by the other submatrices

*This work was supported in part by the National Science Foundation grant CCF-1552323 and by the Applied Mathematics Program within the Office of Science of the U.S. Department of Energy by grant DE-SC0010205 at Purdue, and at the Pacific Northwest National Laboratory, operated by Battelle for the DOE under Contract DE-AC05-76RL01830.

†Department of Computer Science, Purdue University, West Lafayette, IN (yyeung@purdue.edu, apothen@purdue.edu).

‡Pacific Northwest National Laboratory, Richland, WA (hala@pnnl.gov, Zhenyu.Huang@pnnl.gov).

of the block matrix. We describe two algorithms to solve this augmented system. In both algorithms, the original matrix A is factored with a direct method. In the first algorithm, the Schur complement system is also solved by a direct method, and in the second algorithm it is solved with a Krylov subspace solver. We maintain symmetry in the augmented system of equations and the second algorithm whereas in the first algorithm an unsymmetric system is solved to reduce the computation time. Note that our algorithms can handle arbitrary changes to $\hat{\boldsymbol{b}}$ in (2). However, in the power grid application considered here, $\hat{\boldsymbol{b}}$ only changes in the set of m rows where the principal submatrix is updated. Hence we focus on this situation in our experiments.

Our motivation for this work comes from dynamically assessing the security of power grids, which is also called contingency analysis. In power engineering, an interconnected power system is described by a system of complex, nonlinear equations representing the relationship between voltages, powers and admittances at points of interest called buses. Here, we consider the "DC" approximation of this problem, which is derived using heuristic assumptions, and is described by a linear system,

$$-B\boldsymbol{d} = \boldsymbol{p}, \tag{3}$$

where B is the imaginary component of the $n \times n$ admittance matrix, $\boldsymbol{d}$ is an n-vector of the relative phase shift of the voltage, $\boldsymbol{p}$ is an n-vector of the real power, and n is the number of buses in the system. In contingency analysis, one removes an existing connection between two buses in the system to simulate the failure to transmit power through that transmission line, or all the connections to a generator to simulate the failure to generate power from it. Removing a connection in the system corresponds to a principal submatrix update to (3), and the updated matrix has the same size n as the original matrix. Bienstock discusses a mixed-integer programming approach to this problem [1], which restricts the size of the problems they can solve to a few hundred buses.

We propose AMPS, an augmented system that is equivalent to (2), which means in exact arithmetic solving the augmented system would give us the same solution vector $\hat{\boldsymbol{x}}$. Our experimental results show that the accuracy of the solution to the augmented system is comparable to that of the solution $\hat{\boldsymbol{x}}$ obtained by solving (2) by a direct method.

Our algorithm satisfies the following four desiderata:

1. The solution of the augmented system should be computed in a number of operations proportional to the size of the update m rather than the size of the system n. This is especially important for large systems when there is a need for a sequence of updates in real-time.
2. The accuracy of the solution to the augmented system should be comparable to that of the direct solution of the modified system.
3. Both the factors of the matrix and the solution of the original system should be utilized in solving the augmented system to avoid redundant computations.
4. Sparsity in the matrices and the vectors should be exploited to accelerate the computations.

The work most closely related to this paper is an augmented matrix approach to solving the stiffness system of equations in a surgery stimulation when an organ is cut or deformed, proposed by Yeung, Crouch and Pothen [16]. The surgery is visualized by updating a finite element formulation of a linear elastic model of the organ as it is cut. The matrix here is the (varying) stiffness matrix from the finite element model of the organ. For surgery simulations, solutions of tens or hundreds

of modified systems per second are needed. With the augmented matrix approach, the stiffness matrix of the initial mesh can be kept unchanged, and all changes as the mesh is being cut can be described using the $(1,2)$- and $(2,1)$-blocks of a block 2×2 matrix. In this problem, nodes and elements could be deleted, added, or replaced, and thus the dimension of the matrix changes, unlike the situation here. These authors used an unsymmetric form of the augmented matrix with a hybrid direct-iterative algorithm, where a direct method was used to factor the initial stiffness matrix, and the Schur complement system was solved implicitly using a Krylov space solver. There are two major differences here. The first is that the update is restricted to a principal submatrix in the power grid context. The second is that symmetry is preserved while it was destroyed in the earlier method even though both the matrix and the updates were symmetric. There are other existing augmented matrix approaches, which will be discussed later in this paper.

Notation. We use Householder notation throughout; that is, matrices are denoted by upper case Roman letters, vectors by lower case Roman letters, and scalars by Greek letters. There are some exceptions: Indices and dimensions are also denoted by lower case Roman letters (e.g. i, j, k and m, n). With this convention, the elements of a matrix A are denoted by α_{ij}, and the elements of a vector $\boldsymbol{x}$ are denoted by χ_j. A submatrix of A is denoted by A_{ij}, and a subvector of $\boldsymbol{x}$ is denoted by $\boldsymbol{x}_j$. We use $A^\top$ to denote the transpose of A. The symbols L and D are reserved for lower triangular and (block) diagonal matrices. The jth column of the identity matrix I is written as $\boldsymbol{e}_j$, and thus the matrix H in (1) is $H = [\boldsymbol{e}_{j_1}, \boldsymbol{e}_{j_2}, \ldots, \boldsymbol{e}_{j_m}]$ for the set of indices of the modified rows and columns $\mathbb{S} = \{j_1, j_2, \ldots, j_m\}$.

Organization of this article. Section 2 presents our new augmented system of equations for solving the modified system when a principal submatrix is updated. Section 3 describes the details of the algorithm to solve the modified system using the augmented formulation presented in the previous section. Section 4 presents computational times and the accuracy of solutions when the augmented system is applied to contingency analysis of three power grids. Section 5 discusses conclusions and directions for future work.

2. Augmented system formulation. It is well known that augmented systems can be used to effectively add and remove rows and columns of matrices [2, 7]. We begin by describing how these operations are accomplished, assuming that both the original matrix and the modifications are symmetric, i.e., the procedures are applied to rows and columns simultaneously. These modifications are not restricted to principal submatrix updates. Also, these modifications might not preserve the nonsingularity of the matrix. Hence after each update, we characterize the conditions that must be satisfied for the updated matrix to be nonsingular when the initial matrix is nonsingular. These results are obtained using the determinantal identity

$$\det \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \det(A) \det(D - CA^{-1}B),$$

when A is nonsingular. The goal of these characterizations is to show that our augmented system formulation by itself does not create singular matrices.

2.1. Adding a row and a column. To add a row and a column to $A\boldsymbol{x} = \boldsymbol{b}$, we consider the system

$$\begin{bmatrix} A & \hat{\boldsymbol{a}} \\ \hat{\boldsymbol{a}}^\top & \hat{\alpha} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}}_1 \\ \hat{\chi} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ \hat{\beta} \end{bmatrix}. \tag{4}$$

If A is nonsingular and $\hat{\alpha} \neq \hat{\boldsymbol{a}}^\top A^{-1}\hat{\boldsymbol{a}}$, then the augmented matrix is nonsingular; and if A is positive definite and $\hat{\alpha} > \hat{\boldsymbol{a}}^\top A^{-1}\hat{\boldsymbol{a}}$, then the augmented matrix is also positive definite.

2.2. Removing a row and a column. To remove the jth row and column from $A\boldsymbol{x} = \boldsymbol{b}$, we consider the system

$$\begin{bmatrix} A & \boldsymbol{e}_j \\ \boldsymbol{e}_j^\top & 0 \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}}_1 \\ \hat{\chi} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ 0 \end{bmatrix}. \tag{5}$$

The last row $\boldsymbol{e}_j^\top \hat{\boldsymbol{x}}_1 = 0$ constrains the jth component of $\hat{\boldsymbol{x}}_1$ to be 0, and consequently removes the contribution of the jth column of A. This leaves us with one fewer *effective* variable than the number of equations. This is compensated by the additional component $\hat{\chi}$ in the solution vector. Consider the jth row of the augmented system: $\boldsymbol{e}_j^\top A\hat{\boldsymbol{x}}_1 + \hat{\chi} = \boldsymbol{e}_j^\top \boldsymbol{b}$. Since $\hat{\chi}$ only appears in the j^{th} row of the system, it is constrained to the value $\boldsymbol{e}_j^\top(\boldsymbol{b} - A\hat{\boldsymbol{x}}_1)$ after all the other components of $\hat{\boldsymbol{x}}_1$ are determined. Its value will be discarded after the system is solved.

Augmentation in this manner makes the matrix symmetric indefinite. If A is a symmetric positive definite matrix, then we can show that the augmented matrix is nonsingular, since its determinant is equal to $-\det(A)\,(A^{-1})_{jj}$, and both terms are positive.

2.3. Replacing a row and a column. Replacing a row and a column can be done by removing the old row and column and adding the new ones. The resulting augmented formulation would be

$$\begin{bmatrix} A & \hat{\boldsymbol{a}}_j & \boldsymbol{e}_j \\ \hat{\boldsymbol{a}}_j^\top & \hat{\alpha}_{jj} & 0 \\ \boldsymbol{e}_j^\top & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}}_1 \\ \hat{\chi}_1 \\ \hat{\chi}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ \hat{\beta} \\ 0 \end{bmatrix}, \tag{6}$$

where $\hat{\boldsymbol{a}}_j$ and $\hat{\alpha}_{jj}$ are the jth column and the (j,j)th element of $\hat{A}$ in (2) respectively. Note that the jth component of $\hat{\boldsymbol{a}}_j$ is then multiplied by the jth component of $\hat{\boldsymbol{x}}_1$ which is constrained to be 0 by the last equation. Hence the jth component of $\hat{\boldsymbol{a}}_j$ can be chosen arbitrarily.

We can calculate the determinant of the augmented matrix as

$$\det(A)\ \det \begin{bmatrix} \hat{\alpha}_{jj} - \hat{\boldsymbol{a}}_j^\top A^{-1}\hat{\boldsymbol{a}}_j & -\hat{\boldsymbol{a}}_j^\top A^{-1}\boldsymbol{e}_j \\ -\boldsymbol{e}_j^\top A^{-1}\hat{\boldsymbol{a}}_j & -\boldsymbol{e}_j^\top A^{-1}\boldsymbol{e}_j \end{bmatrix}.$$

Hence if A and the 2×2 matrix above are both nonsingular, the augmented matrix is also nonsingular.

2.4. Replacing multiple rows and columns. Replacing m rows and columns can be done by concatenating the replaced rows and columns. Suppose the set of indices of the rows and columns to be replaced is $\mathbb{S} = \{j_1, j_2, \ldots, j_m\}$. The complete augmented formulation would be

$$\begin{bmatrix} A & J & H \\ J^\top & C & 0 \\ H^\top & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}}_1 \\ \hat{\boldsymbol{x}}_2 \\ \hat{\boldsymbol{x}}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ H^\top \hat{\boldsymbol{b}} \\ \boldsymbol{0} \end{bmatrix}, \tag{7}$$

where $J = [\hat{\boldsymbol{a}}_{j_1}, \hat{\boldsymbol{a}}_{j_2}, \ldots, \hat{\boldsymbol{a}}_{j_m}]$ are the modified columns of $\hat{A}$, $H = [\boldsymbol{e}_{j_1}, \boldsymbol{e}_{j_2}, \ldots, \boldsymbol{e}_{j_m}]$ is the submatrix of the identity matrix with the indices of the columns to be replaced,

and $C = H^\top \hat{A} H$ is the diagonal block of the modified matrix $\hat{A}$ where the changes occur. Note that the submatrix C is $m \times m$, J and H are $n \times m$, and $m \ll n$.

Again, if A is nonsingular, we can express the determinant of the augmented matrix as

$$\det(A)\ \det \begin{bmatrix} C - J^\top A^{-1} J & -J^\top A^{-1} H \\ -H^\top A^{-1} J & -H^\top A^{-1} H \end{bmatrix}.$$

If A and the second matrix above are both nonsingular, then the augmented matrix is also nonsingular. (We can choose $J = AH$ as shown later in this section; then the block 2×2 matrix above is the negation of the Schur complement matrix S_1 in the iterative variant of our AMPS algorithm in Section 3.) In other words, the augmented matrix is nonsingular if and only if both A and the Schur complement matrix S_1 are nonsingular.

We proceed to refine the system of equations (7) further. With a suitable $n \times n$ permutation matrix P, we can partition H into an identity matrix and a zero matrix:

$$PH = \begin{bmatrix} I_m \\ 0_{n-m} \end{bmatrix}. \tag{8}$$

Applying the same permutation matrix P to J, A, $\hat{\boldsymbol{x}}_1$ and $\boldsymbol{b}$ yields

$$PJ = \begin{bmatrix} J_1 \\ J_2 \end{bmatrix}, \qquad PAP^\top = \begin{bmatrix} A_{11} & A_{12} \\ A_{12}^\top & A_{22} \end{bmatrix}, \tag{9a}$$

$$P\hat{\boldsymbol{x}}_1 = \begin{bmatrix} \hat{\boldsymbol{x}}_{11} \\ \hat{\boldsymbol{x}}_{12} \end{bmatrix}, \qquad P\boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{bmatrix}. \tag{9b}$$

We can then apply the permutation matrix

$$\hat{P} = \begin{bmatrix} P & & \\ & I_m & \\ & & I_m \end{bmatrix} \tag{10}$$

to the matrix in (7) from both left and right, which yields

$$\begin{bmatrix} A_{11} & A_{12} & J_1 & I \\ A_{12}^\top & A_{22} & J_2 & 0 \\ J_1^\top & J_2^\top & C & 0 \\ I & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}}_{11} \\ \hat{\boldsymbol{x}}_{12} \\ \hat{\boldsymbol{x}}_2 \\ \hat{\boldsymbol{x}}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ H^\top \hat{\boldsymbol{b}} \\ \boldsymbol{0} \end{bmatrix}. \tag{11}$$

Here A_{11} is the $m \times m$ submatrix being replaced by C, A_{22} is the $(n-m) \times (n-m)$ principal submatrix of A that is unchanged, and A_{12} is the $m \times (n-m)$ off-diagonal submatrix of A. Note that the third column block effectively replaces the first column block, and by symmetry in the update, the third row block also replaces the first row block. Hence, the submatrix $\begin{bmatrix} J_2 \\ C \end{bmatrix}$ must consist of the modified columns in $\hat{A}$ that correspond to the original columns $\begin{bmatrix} A_{12}^\top \\ A_{11} \end{bmatrix}$ in A.

LEMMA 1. *The submatrix J_1 in (11) can be chosen arbitrarily such that the system is always consistent. Moreover, if $\hat{A}$ in (2) is nonsingular, $\hat{\boldsymbol{x}}_{12}$ and $\hat{\boldsymbol{x}}_2$ are independent of the submatrix J_1.*

Proof. Consider the last row block of (11). We have $\hat{\boldsymbol{x}}_{11} = \mathbf{0}$. Consequently, the first column block, which then multiplies $\hat{\boldsymbol{x}}_{11}$, does not contribute to the solution of the system. Moreover, consider the first row block of (11):

$$A_{12}\hat{\boldsymbol{x}}_{12} + J_1\hat{\boldsymbol{x}}_2 + \hat{\boldsymbol{x}}_3 = \boldsymbol{b}_1. \tag{12}$$

Since $\hat{\boldsymbol{x}}_3$ only contributes to one row block in the system of equations, its values can be determined uniquely for any values of J_1. Hence the submatrix J_1 can be chosen arbitrarily.

Now we can prove the second statement in the lemma. If we consider the second and third row and column blocks of system (11), since the last column blocks are zero for these rows, we have, after row and column permutations,

$$\begin{bmatrix} C & J_2^\top \\ J_2 & A_{22} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}}_2 \\ \hat{\boldsymbol{x}}_{12} \end{bmatrix} = \begin{bmatrix} H^\top \hat{\boldsymbol{b}} \\ \boldsymbol{b}_2 \end{bmatrix}. \tag{13}$$

(This system is the updated $n \times n$ system of equations (2) written in its block 2×2 form.) Hence the vectors $\hat{\boldsymbol{x}}_{12}$ and $\hat{\boldsymbol{x}}_2$ are independent of the submatrix J_1. ∎

Note that the values of $\hat{\boldsymbol{x}}_2$ and $\hat{\boldsymbol{x}}_3$ are coupled in (12), i.e., we can express one in terms of the other. Therefore, we need only one of them when solving the updated solution $\hat{\boldsymbol{x}}$ in (2).

Since we have applied the permutation to the solution vector in (11), we need to unpermute it to obtain the updated solution $\hat{\boldsymbol{x}}$ in (2). Hence we obtain

$$P\hat{\boldsymbol{x}} = \begin{bmatrix} \hat{\boldsymbol{x}}_2 \\ \hat{\boldsymbol{x}}_{12} \end{bmatrix}. \tag{14}$$

2.5. Principal submatrix update. We now extend the techniques described in the previous subsection to design an augmented matrix approach to update the solution when A is modified by a principal submatrix update as in (1). In this case, all the changes are captured in the submatrix C in (7), and we can deduce that $C = H^\top \hat{A} H = H^\top A H - E$. Therefore, the submatrix J_2 in (11) remains unchanged from the original system and thus $J_2 = A_{12}^\top$. As proven in the previous subsection, J_1 in (11) can be chosen arbitrarily. With the choice of $J_1 = A_{11}$, we can show that

$$J = P^\top \begin{bmatrix} J_1 \\ J_2 \end{bmatrix} = P^\top \begin{bmatrix} A_{11} \\ A_{12}^\top \end{bmatrix} = AH. \tag{15}$$

The last equation follows from

$$AH = P^\top \begin{bmatrix} A_{11} & A_{12} \\ A_{12}^\top & A_{22} \end{bmatrix} PH = P^\top \begin{bmatrix} A_{11} & A_{12} \\ A_{12}^\top & A_{22} \end{bmatrix} \begin{bmatrix} I_m \\ 0_{n-m} \end{bmatrix} = P^\top \begin{bmatrix} A_{11} \\ A_{12}^\top \end{bmatrix}. \tag{16}$$

We can thus write (7) as

$$\begin{bmatrix} A & AH & H \\ H^\top A & C & 0 \\ H^\top & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}}_1 \\ \hat{\boldsymbol{x}}_2 \\ \hat{\boldsymbol{x}}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ H^\top \hat{\boldsymbol{b}} \\ \mathbf{0} \end{bmatrix}. \tag{17}$$

Here is an example in which the principal submatrix at the 3rd and 5th rows and

columns are modified. The augmented system (17) would be

$$
\left[\begin{array}{cccccc|cccc}
 & & & & & & \alpha_{13} & \alpha_{15} & 0 & 0 \\
 & & & & & & \alpha_{23} & \alpha_{25} & 0 & 0 \\
 & & & A & & & \underline{\alpha_{33}} & \underline{\alpha_{35}} & 1 & 0 \\
 & & & & & & \alpha_{43} & \alpha_{45} & 0 & 0 \\
 & & & & & & \underline{\alpha_{53}} & \underline{\alpha_{55}} & 0 & 1 \\
 & & & & & & \vdots & \vdots & \vdots & \vdots \\
\hline
\alpha_{31} & \alpha_{32} & \underline{\alpha_{33}} & \alpha_{34} & \underline{\alpha_{35}} & \cdots & \hat{\alpha}_{33} & \hat{\alpha}_{35} & 0 & 0 \\
\alpha_{51} & \alpha_{52} & \underline{\alpha_{53}} & \alpha_{54} & \underline{\alpha_{55}} & \cdots & \hat{\alpha}_{53} & \hat{\alpha}_{55} & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & \cdots & 0 & 0 & 0 & 0
\end{array}\right]
\left[\begin{array}{c} \hat{\chi}_1 \\ \hat{\chi}_2 \\ \zeta_3 \\ \hat{\chi}_4 \\ \zeta_5 \\ \vdots \\ \hline \hat{\chi}_3 \\ \hat{\chi}_5 \\ \delta_3 \\ \delta_5 \end{array}\right]
= \left[\begin{array}{c} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \vdots \\ \hline \hat{\beta}_3 \\ \hat{\beta}_5 \\ 0 \\ 0 \end{array}\right], \tag{18}
$$

in which the ζ terms are constrained to be 0, the χ terms are the permuted solutions to (2), and the δ terms are the values of $\hat{\boldsymbol{x}}_3$ in (17).

3. Solution method. In this section, we describe our algorithms to solve the system (7). Suppose we have computed the $\mathrm{LDL}^\top$ factorization of A when solving the original system $A\boldsymbol{x} = \boldsymbol{b}$. Here, L is a unit lower triangular matrix and D is a diagonal matrix or block diagonal matrix with 1×1 or 2×2 blocks if A is indefinite. A fill-reducing ordering and an ordering to maintain numerical stability are usually used during the factorization, and thus a permuted matrix of A is factored, i.e., $\dot{P}^\top A \dot{P} = LDL^\top$ for some permutation matrix $\dot{P}$. We assume that hereafter the permuted system $\dot{P}^\top A \dot{P} \dot{P}^\top \boldsymbol{x} = \dot{P}^\top \boldsymbol{b}$ has replaced the original system. Solutions to the original system $A\boldsymbol{x} = \boldsymbol{b}$ can then be obtained by applying the inverse permutation $\dot{P}$. For simplicity, we will not explicitly write the permutation matrices $\dot{P}$. We can solve (17) in two ways.

Iterative method. With $A = LDL^\top$ as a block pivot, (17) can be reduced to a smaller system involving the symmetric matrix S_1, the Schur complement of A, with a multiplication by -1:

$$
\underbrace{\begin{bmatrix} E & I \\ I & H^\top A^{-1} H \end{bmatrix}}_{S_1} \begin{bmatrix} \hat{\boldsymbol{x}}_2 \\ \hat{\boldsymbol{x}}_3 \end{bmatrix} = \begin{bmatrix} H^\top (\boldsymbol{b} - \hat{\boldsymbol{b}}) \\ H^\top A^{-1} \boldsymbol{b} \end{bmatrix}, \tag{19}
$$

where $E = H^\top A H - C$, which is the same E as in (1). This can be shown by premultiplying and postmultiplying (1) by $H^\top$ and H respectively:

$$
C \equiv H^\top \hat{A} H = H^\top A H - H^\top H E H^\top H = H^\top A H - E. \tag{20}
$$

We can solve (19) by an iterative method such as GMRES or MINRES. Matrix-vector products with S_1 need a partial solve with $A = LDL^\top$ involving only the rows and columns selected by H and $H^\top$, and products with E, in each iteration. Note that $H^\top$ in the right-hand-side vector selects the components from the difference vector $\hat{\boldsymbol{b}} - \boldsymbol{b}$ and the solution $\boldsymbol{x}$ of the original system $A\boldsymbol{x} = \boldsymbol{b}$ if the changes in the right-hand-side vector are only in the changed rows of A.

Direct method. Alternatively, the $(2,1)$-block of S_1 can be used as a block pivot with another Schur complement:

$$
\big(\underbrace{E H^\top A^{-1} H - I}_{S_2}\big) \hat{\boldsymbol{x}}_3 = E H^\top A^{-1} \boldsymbol{b} - H^\top (\boldsymbol{b} - \hat{\boldsymbol{b}}). \tag{21}
$$

We can then solve this equation for $\hat{\boldsymbol{x}}_3$ using a direct solver with an LU factorization of S_2, which can be constructed efficiently as described later in subsection 3.3. Note that S_2 is unsymmetric although both the augmented matrix in (17) and S_1 in (19) are symmetric. This is because we have chosen an off-diagonal block pivot in forming S_2, to avoid the computation of the inverse of either E or $H^\top A^{-1} H$ on the diagonal block of S_1.

3.1. Solution to the modified system. It turns out that we only need to compute $\hat{\boldsymbol{x}}_3$ to obtain the full solution vector $\hat{\boldsymbol{x}}$ to the modified system (2). This can be done by making the following observation. Premultiplying the first row block of (17) by A^{-1}, and rearranging terms yields

$$\hat{\boldsymbol{x}}_1 = A^{-1}\boldsymbol{b} - H\hat{\boldsymbol{x}}_2 - A^{-1}H\hat{\boldsymbol{x}}_3. \tag{22}$$

From (14), we have

$$P\hat{\boldsymbol{x}} = \begin{bmatrix} \hat{\boldsymbol{x}}_2 \\ \hat{\boldsymbol{x}}_{12} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{x}}_{11} \\ \hat{\boldsymbol{x}}_{12} \end{bmatrix} + \begin{bmatrix} \hat{\boldsymbol{x}}_2 \\ \boldsymbol{0} \end{bmatrix} = P\hat{\boldsymbol{x}}_1 + PH\hat{\boldsymbol{x}}_2, \tag{23}$$

by using (8) and (9b), and the fact that $\hat{\boldsymbol{x}}_{11} = \boldsymbol{0}$. Premultiplying both sides by $P^\top$ yields

$$\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}_1 + H\hat{\boldsymbol{x}}_2. \tag{24}$$

Substituting (22) into (24), we have

$$\hat{\boldsymbol{x}} = A^{-1}\boldsymbol{b} - A^{-1}H\hat{\boldsymbol{x}}_3, \tag{25}$$

in which the first term is the solution to the original system.

3.2. Relation to the Sherman-Morrison-Woodbury formula. The solution $\hat{\boldsymbol{x}}$ in (2) obtained by AMPS using the direct approach can be expressed in a single equation by substituting $\hat{\boldsymbol{x}}_3$ in (21) into (25):

$$\hat{\boldsymbol{x}} = A^{-1}\boldsymbol{b} - A^{-1}H\left(EH^\top A^{-1}H - I\right)^{-1}\left[EH^\top A^{-1}\boldsymbol{b} - H^\top(\boldsymbol{b} - \hat{\boldsymbol{b}})\right]. \tag{26}$$

In the case when the right-hand-side of (2) does not change from the original system, i.e. $\hat{\boldsymbol{b}} = \boldsymbol{b}$, (26) becomes

$$\hat{\boldsymbol{x}} = \left[A^{-1} - A^{-1}H\left(EH^\top A^{-1}H - I\right)^{-1}EH^\top A^{-1}\right]\boldsymbol{b}. \tag{27}$$

Using the Sherman-Morrison Woodbury formula

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + VA^{-1}U\right)^{-1}VA^{-1}, \tag{28}$$

the inverse of $\hat{A}$ in (1) can be expressed as

$$\begin{aligned} \hat{A}^{-1} &= \left[A + (H)\,(-I)\left(EH^\top\right)\right]^{-1} \\ &= A^{-1} - A^{-1}H\left(-I + EH^\top A^{-1}H\right)^{-1}EH^\top A^{-1}, \end{aligned} \tag{29}$$

which when multiplied by $\boldsymbol{b}$ is identical to (27).

3.3. Forming the Schur complement S_2 explicitly. In the previous subsection we have described the algorithm to solve the modified system using the augmented formulation. We now discuss how we form the matrix $W \equiv EH^\top A^{-1}H$ in the Schur complement S_2 in (21) by using partial triangular solves.

From the factorization of A, the matrix $H^\top A^{-1}H$ can be expressed as the product $H^\top L^{-\top}D^{-1}L^{-1}H$. Then $W^\top$ can be expressed as

$$W^\top = H^\top L^{-\top}D^{-1}L^{-1}HE. \tag{30}$$

Recall that E is symmetric. Let $X \equiv L^{-1}HE$. Premultiplying both sides of this equation by L, we have

$$LX = HE \equiv \tilde{E}. \tag{31}$$

Observe that the right-hand-side of (31) is a matrix $\tilde{E}$ mapping the i^{th} row of E to the j_i^{th} row of $\tilde{E}$ with the rest of $\tilde{E}$ filled with 0. For instance, if the set of indices of updates is $\mathbb{S} = \{3, 5\}$, then (31) would be

$$LX = HE = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ \vdots & \vdots \end{bmatrix} E = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \epsilon_{11} & \epsilon_{12} \\ 0 & 0 \\ \epsilon_{21} & \epsilon_{22} \\ \vdots & \vdots \end{bmatrix}. \tag{32}$$

Since both L and $\tilde{E}$ are sparse, we can use partial forward substitution to solve for X.

Let $Y \equiv L^{-1}H$. We can again use partial forward substitution by exploiting the sparsity in H and L to compute Y, as discussed in the next subsection. Once we have the matrices X and Y, we can compute $W^\top$ as follows:

$$W^\top = H^\top L^{-\top}D^{-1}X = Y^\top D^{-1}X. \tag{33}$$

3.3.1. Exploiting sparsity in the computations. To describe how we exploit the sparsity in the matrices and the vectors to reduce the complexity of our algorithms, we need a few concepts from sparse matrix theory as outlined below. We start with a few definitions to help with the discussions that follow.

DEFINITION 2. *An $n \times n$ sparse matrix A can be represented by a directed graph $G(A)$ whose vertices are the integers $1, \ldots, n$ and whose edges are*

$$\{(i, j) : i \neq j, \text{ and } \alpha_{ij} \neq 0\}.$$

The edge (i, j) is directed from vertex i to j. The set of edges is also called the (nonzero) structure of A.

The transitive reduction of a directed graph $G = (V, E)$ is obtained by deleting from the set of edges E every edge (i, j) such that there is a directed path from vertex i to j that does not use the edge (i, j) itself.

DEFINITION 3. *An elimination tree of a Cholesky factor L is the transitive reduction of the directed graph $G(L)$. [11]*

DEFINITION 4. *The (nonzero) structure of an n-vector* $\boldsymbol{x}$ *is*

$$\text{struct}\,(x) := \{i : \chi_i \neq 0\},$$

which can be interpreted as a set of vertices of the directed graph of any $n \times n$ *matrix. In this paper, for a vector* $\boldsymbol{x}$, $\text{closure}_A\,(\boldsymbol{x})$ *refers to* $\text{closure}_A\,(\text{struct}\,(\boldsymbol{x}))$.

DEFINITION 5. *Given a directed graph* $G(A)$ *and a subset of its vertices denoted by* V, *we say* V *is closed with respect to* A *if there is no edge of* $G(A)$ *that joins a vertex not in* V *to a vertex in* V; *that is,* $\nu_j \in V$ *and* $\alpha_{ij} \neq 0$ *implies* $\nu_i \in V$. *The closure of* V *with respect to* A *is the smallest closed set containing* V,

$$\text{closure}_A\,(V) := \bigcap\{U : V \subseteq U, \text{ and } U \text{ is closed}\},$$

which is the set of vertices of $G(A)$ *from which there are directed paths in* $G(A)$ *to vertices in* V.

To compute the structure of X in (31), we apply the following theorem.

THEOREM 6. *Let the structures of* A *and* $\boldsymbol{b}$ *be given. Whatever the values of the nonzeros in* A *and* $\boldsymbol{b}$, *if* A *is nonsingular then*

$$\text{struct}\left(A^{-1}\boldsymbol{b}\right) \subseteq \text{closure}_A\,(\boldsymbol{b})\,.$$

The proof of Theorem 6 is due to Gilbert [6]. Hence the structure of each column of X would be the closure of the nonzeros of the corresponding column of $\tilde{E}$ in the graph of $G(L)$. Similarly, to compute the submatrix of $L^\top$ necessary to obtain the needed components of $\tilde{W}$, we can apply the following theorem.

THEOREM 7. *Suppose we need only some of the components of the solution vector* $\boldsymbol{x}$ *of the system* $A\boldsymbol{x} = \boldsymbol{b}$. *Denote the needed components by* $\tilde{\boldsymbol{x}}$. *If* A *is nonsingular, then the set of components in* $\boldsymbol{b}$ *needed is* $\text{closure}_{A^\top}\,(\tilde{\boldsymbol{x}})$.

The proof of Theorem 7 can be found in [16]. Hence we can deduce that if $\mathbb{S}$ is the set of indices of updates, the submatrix of $L^\top$ needed would also be the closure of $\mathbb{S}$, which is the same as the row indices of nonzeros in the columns of X. (Recall that $\mathbb{S}$ has cardinality m.) Since L is a Cholesky factor, this closure is equivalent to the union of all the vertices on the paths from $\mathbb{S}$ to the root of the elimination tree of $G(L)$, denoted as $P_{\mathbb{S}}$, as proven among others in [16]. We denote the size of this closure by ρ:

$$\rho \equiv |\,\text{closure}_L\,(\mathbb{S})\,| = \sum_{k \in P_{\mathbb{S}}} |L_{*k}|. \tag{34}$$

The upper bound on ρ is the total number of nonzeros in L, denoted as $|L|$. In practice since $m \ll n$, this upper bound is quite loose, and ρ is closer to a small constant times m than to $|L|$.

3.4. Complexity analysis. The time complexity of principal submatrix updates using the symmetric augmented formulation can be summarized in Table 1. Direct method refers to the approach of solving for $\hat{\boldsymbol{x}}_3$ directly using (21), and iterative method refers to the approach of applying an iterative method to (19). Recall that n is the size of the original matrix A, m is the size of the principal submatrix update C, while t denotes the number of iterations that the iterative method takes to converge.

	Computation	Complexity	
		Direct method	Iterative method
Amortized initialization:			
1	Compute $\mathrm{LDL}^\top$ factorization of A	$O(n^{3/2})$ for planar networks	
2	Compute $\boldsymbol{x} = A^{-1}\boldsymbol{b}$	$O(\lvert L\rvert)$	
Real-time update steps:			
1	Obtain the submatrix B	$O(\lvert B\rvert) \le O(m^2)$	
2	Compute $E = H^\top AH - B$	$O(\lvert E\rvert) \le O(m^2)$	
3	Compute $W^\top = H^\top A^{-1}HE$	$O(m \cdot \rho)$	-
	(a) Form $\tilde{E} = HE$	$O(\lvert E\rvert)$	-
	(b) Solve $LX = \tilde{E}$	$O(m \cdot \rho)$	-
	(c) Solve $L^\top \tilde{W}^\top = D^{-1}X$	$O(m \cdot \rho)$	-
	(d) Form $W^\top = H^\top \tilde{W}^\top$	$O(m^2)$	-
4	Form $W - I$	$O(m)$	-
5	Form R.H.S. of (19) and (21)	$O(m)$	$O(\lvert E\rvert + m)$
6	Solve for $\hat{\boldsymbol{x}}_3$	$O(m^3)$	$O(t \cdot \rho)$
7	Solve $\hat{\boldsymbol{x}} = \boldsymbol{x} - A^{-1}H\hat{\boldsymbol{x}}_3$	$O(\lvert L\rvert)$	

TABLE 1
Summary of time complexity

The overall time complexity of the direct method is dominated by either Step 3 (computing $W^\top$) or Step 7 (solving for $\hat{\boldsymbol{x}}$), i.e., $O(m\cdot\rho+|L|)$. For the iterative method, the time complexity is dominated by either Step 6 (solving for $\hat{\boldsymbol{x}}_3$) or Step 7 (solving for $\hat{\boldsymbol{x}}$), i.e., $O(t \cdot \rho + |L|)$. Hence the AMPS algorithms have the time complexities

$$O(m \cdot \rho + |L|) \text{ (direct)} \quad \text{and} \quad O(t \cdot \rho + |L|) \text{ (iterative)}. \tag{35}$$

In comparison, for CHOLMOD [4], the time complexity for updating the Cholesky factor of the matrix, when row and column changes are made, is

$$O\left(\sum_{j\in\mathbb{S}}\left(\sum_{k:\overline{L}_{jk}\neq 0}|L_{*k}| + \sum_{k\in\overline{P}_j}|\overline{L}_{*k}|\right)\right), \tag{36}$$

where L is the original Cholesky factor, $\overline{L}$ is the modified Cholesky factor and $\overline{P}_j$ is the path from node j to the root of the elimination tree of $\overline{L}$. (Note that we have to add the cost $|\overline{L}|$ to compute the solution by solving the triangular system of equations.)

Consider the two inner sums in the expression for the complexity. The first inner sum computes the total number of operations of Steps 1–4 in both Algorithms 1 (Row Addition) and 2 (Row Deletion) in CHOLMOD. If we denote $\mathbb{T}_j$ as the set of nodes $k < j$ in $G(\hat{A})$ that have an edge incident on node j, then this sum is equivalent to the number of outgoing edges of the closure of $\mathbb{T}_j$ in $G(\overline{L})$ up to node j.

The second inner sum computes the number of operations needed for Step 5 (rank-1 update/downdate) in Algorithms 1 and 2 of CHOLMOD. This sum is equivalent to the closure of $\{j\}$ in the updated graph $G(\overline{L})$. Combining the two summation terms, we can express the time complexity of CHOLMOD in terms of the closures:

$$O\left(\sum_{j\in\mathbb{S}} \text{closure}_{\overline{L}}(\mathbb{T}_j)\right) \leq O\left(m \cdot \max_j \text{closure}_{\overline{L}}(\mathbb{T}_j)\right). \tag{37}$$

We make two observations when comparing the AMPS algorithms with CHOLMOD. First, in general, the AMPS algorithms do not introduce new fill-in elements in the Cholesky factor whereas fill-ins are possibly introduced in CHOLMOD. However, this happens when the update introduces a new nonzero entry in row/column j of $\hat{A}$. In our application to the contingency analysis for power flow, we only remove connections between buses. Hence running CHOLMOD neither introduces fill-ins to the factor nor changes the elimination tree. Second, since the nodes in $\mathbb{T}_j$ are numbered less than j, the closure of $\mathbb{T}_j$ is always larger than the closure of $\{j\}$, whether or not the updated factor $\overline{L}$ is different from L. In the case that row j of $\overline{L}$ is relatively dense due to fill-in, the first inner sum in (36) may be the dominant term. On the other hand, the AMPS algorithms only need the closure from node j in $G(L)$.

3.5. Comparison with other augmented methods. Several algorithms have been proposed to solve a modified system of linear equations using augmented matrices. Gill et al. [7] used augmented matrices and a factorization approach to update basis matrices in the simplex algorithm for linear programming, motivated by the work of Bisschop and Meeraus [2, 3]. In their method, the matrix was factored in a block-LU form as

$$\begin{bmatrix} A & \hat{A}H \\ H^\top & \end{bmatrix} = \begin{bmatrix} L & \\ \tilde{Z}^\top & \tilde{D} \end{bmatrix} \begin{bmatrix} U & \tilde{Y} \\ & I \end{bmatrix}. \tag{38}$$

Here the matrix L is unit-lower triangular and the matrix U is upper triangular. The matrices $\tilde{Y}$ and $\tilde{Z}$ are $n \times m$ submatrices of the block factors and $\tilde{D}$ is the Schur complement of A.

Maes [12] and Wong [15] used a similar approach to implement active-set QP solvers with symmetric augmentation to solve the Karush-Kuhn-Tucker (KKT) matrices arising from Hessian updates and factored in a block-LU form as

$$\begin{bmatrix} A & V \\ V^\top & \tilde{C} \end{bmatrix} = \begin{bmatrix} L & \\ Z^\top & I \end{bmatrix} \begin{bmatrix} U & Y \\ & \tilde{S} \end{bmatrix}. \tag{39}$$

Here the submatrices Y and Z are $n \times 2m$ submatrices, doubling the size of $\tilde{Y}$ and $\tilde{Z}$ in (38). These submatrices were updated using sparse triangular solves and $\tilde{S}$ was updated using a dense LU-type factorization. Comparing the augmented matrix in (39) with (7), we have

$$V = \begin{bmatrix} AH & H \end{bmatrix} Q^\top \quad \text{and} \quad \tilde{C} = Q \begin{bmatrix} C & 0 \\ 0 & 0 \end{bmatrix} Q^\top, \tag{40}$$

for some permutation matrix Q.

To take advantage of symmetry, Maes and Wong factored the augmented matrix in a symmetric block-LBL$^\top$ form

$$\begin{bmatrix} A & V \\ V^\top & \tilde{C} \end{bmatrix} = \begin{bmatrix} L & \\ Z^\top & I \end{bmatrix} \begin{bmatrix} D & \\ & \tilde{D} \end{bmatrix} \begin{bmatrix} L^\top & Z \\ & I \end{bmatrix}. \tag{41}$$

The major differences between the our methods and the KKT matrix block-LU/block-LBL$^\top$ update method are as follows. We exploited the explicit forms of the submatrices Z and $\tilde{D}$ in the factorization when the update is a principal submatrix. Specifically, if we factor the augmented matrix in (17) as in (41), we have

$$\begin{bmatrix} A & AH & H \\ H^\top A & C & 0 \\ H^\top & 0 & 0 \end{bmatrix} = \hat{L}\hat{D}\hat{L}^\top, \tag{42}$$

where

$$\hat{L} = \begin{bmatrix} L & & \\ H^\top L & I & \\ H^\top L^{-\top} D^{-1} & 0 & I \end{bmatrix} \quad \text{and} \quad \hat{D} = \begin{bmatrix} D & \\ & -S_1 \end{bmatrix}. \tag{43}$$

Here $\hat{L}$ is a lower triangular matrix, $\hat{D}$ is a matrix whose $(1,1)$-block is (block) diagonal and the rest is the negation of the Schur complement S_1 in (19). Combining the results from (40) and (43), we obtain the relationship between the factors in (41) and those in our method:

$$Z^\top = Q \begin{bmatrix} H^\top L \\ H^\top L^{-\top} D^{-1} \end{bmatrix} \quad \text{and} \quad \tilde{D} = Q(-S_1)Q^\top. \tag{44}$$

Hence, we do not need to construct Z and $\tilde{D}$ as in the Gill et al., Maes, and Wong algorithms. We also make use of the structure of the factors in computing the solution, whereas Maes updated the factors by treating the augmentation submatrix V as sparse and $\tilde{C}$ as dense. Finally, we compute the solution to the modified system by explicitly using the solution to the original system.

4. Experimental results. The augmented matrix solution method was evaluated through a series of $N - k$ contingency analyses of two real-world power systems, the 3,120-bus Polish system from the MATPOWER repository [17] and the 14,090-bus WECC system; and a 777,646-bus generated system, which is based on the *case2736sp* system from the MATPOWER repository and the IEEE 123 bus distribution feeder [9]. The distribution feeder is balanced by equivalencing the load on each phase and extending the unbalanced laterals. Several distribution feeders are added at appropriate locations in the transmission case to create this system.

This section provides relevant implementation details and presents experimental results, including comparisons with the PARDISO direct solver [10, 13, 14] on the modified systems, and the CHOLMOD direct solver that updates the factors of A according to $\hat{A}$.

Since the power flow systems follow Kirchhoff's current law, the admittance matrix B in (3) is a weighted Laplacian. A boundary condition is applied to fix the phase shift of a selected bus called the slack bus, and the reduced system is nonsingular but with an eigenvalue close to zero. Hence in the power community a direct solver is usually used to solve the system.

The estimated condition numbers of the admittance matrices B calculated by using MATLAB's `condest` function are 1.2×10^6 for the 3,120-bus Polish system, 2.1×10^7 for the 14,090-bus WECC system, and 9.9×10^8 for the 777,646-bus generated system. The estimated eigenvalues with the smallest magnitude calculated by using MATLAB's `eigs` function are 5.0×10^{-2} for the Polish system, 2.3×10^{-3} for the WECC system, and 1.4×10^{-4} for the generated system.

4.1. Implementation. All experiments were conducted on a desktop computer with four 8-core Intel Xeon E5-2670 processors running at 2.6 GHz with 20 GB cache and 256 GB RAM. All reported times represent the average of 20 runs.

The precomputed $\mathrm{LDL}^\top$ factorizations of the admittance matrices were computed using Oblio, a direct solver library for solving sparse symmetric linear systems of equations with data structure support for dynamic pivoting using 1×1 and 2×2 pivots [5]. Both the GMRES iterative solver used in (19) and the PARDISO solver applied to (2) for comparison purposes were from the Intel Math Kernel Library (MKL) [8]. The CHOLMOD solver applied to (2) was from the SparseSuite package. The remainder of the code was written by the authors.

All matrices were stored in sparse matrix format to reduce both the storage space and access time.

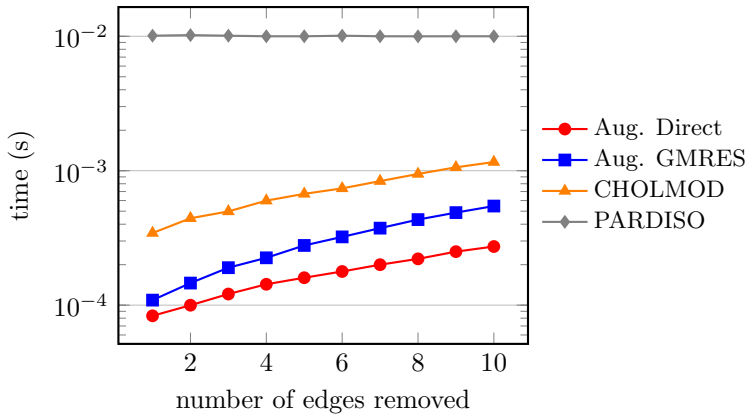
Problem	Aug. Direct	Aug. GMRES	PARDISO	CHOLMOD
3,120-bus Polish system	2×10^{-13}	3×10^{-13}	2×10^{-13}	2×10^{-13}
14,070-bus WECC system	4×10^{-13}	4×10^{-13}	4×10^{-13}	5×10^{-13}
777,646-bus system	6×10^{-12}	6×10^{-12}	6×10^{-12}	5×10^{-12}

TABLE 2
Average relative residual norms ($\|\hat{A}\hat{\boldsymbol{x}} - \hat{\boldsymbol{b}}\|_2 / \|\hat{\boldsymbol{b}}\|_2$) *for each problem*

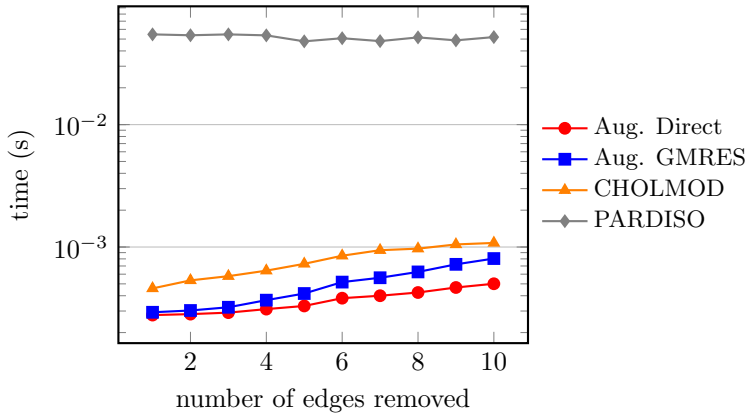
4.2. Experiments. In our $N - k$ contingency analysis experiments we remove k out of N connections in the power grid and form the modified system (2). This corresponds to a principal submatrix update as described in (1), where H is formed by the columns of the identity matrix corresponding to the end-points of the removed connections, and $m \leq 2k$.

We compare the solution of the augmented system using an iterative solver on (19) and using (21) by means of the LU factorization of the Schur complement matrix S_2. Note that since B in (3) is a weighted Laplacian, the update matrix E at the $(1,1)$-block of (19) is singular, and thus the whole matrix is symmetric indefinite. We have used the MINRES method and the generalized minimum residual (GMRES) method to solve these indefinite systems. For MINRES, the average solve time per iteration is faster than the GMRES method, but since it converged slowly and needed more iterations than GMRES, the total solve time was higher than the latter. Hence we report times obtained from GMRES. We also compare our augmented system with PARDISO and CHOLMOD being applied to (2).

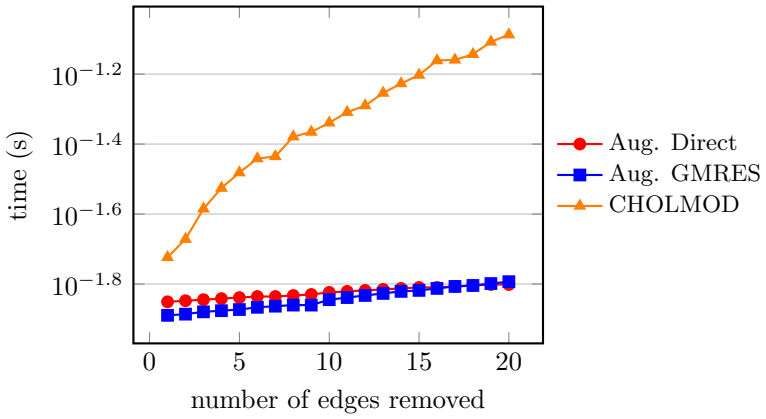
The $\mathrm{LDL}^\top$ factorization times using Oblio were 0.0306 seconds for the 3,120-bus Polish system, 0.156 seconds for the 14,070-bus WECC system, and 2.53 seconds for the 777,646-bus generated system. In comparison, the average factorization times



(a) 3,120-bus Polish system



(b) 14,090-bus WECC system



(c) 777,646-bus generated system

FIG. 1. *Timing results of compared methods*

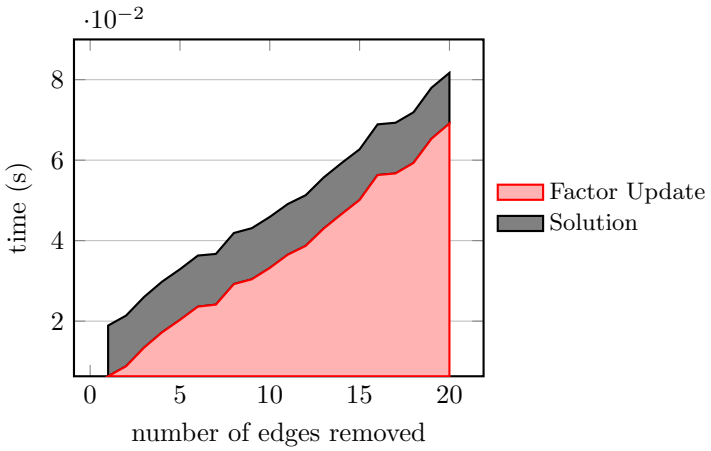
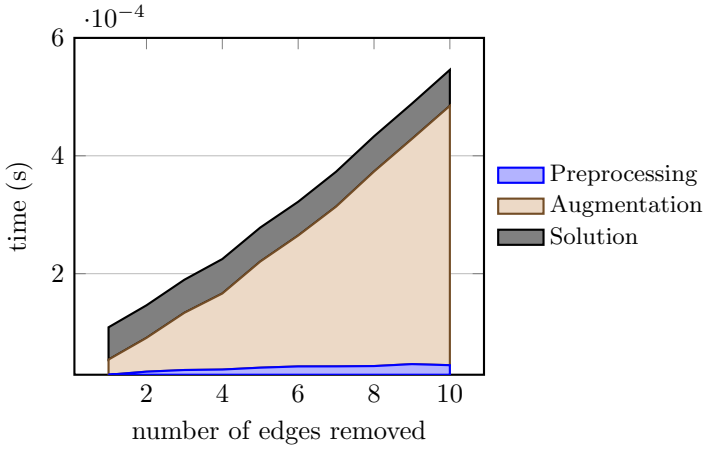
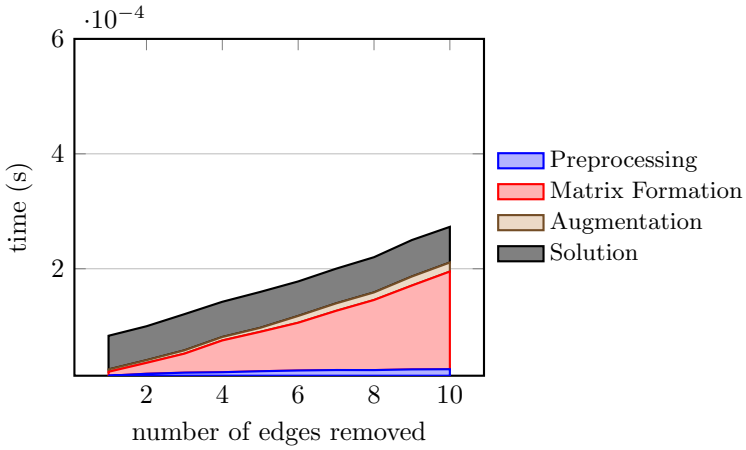


FIG. 2. *Breakdown of the time of the CHOLMOD method for the* 777,646*-bus generated system*

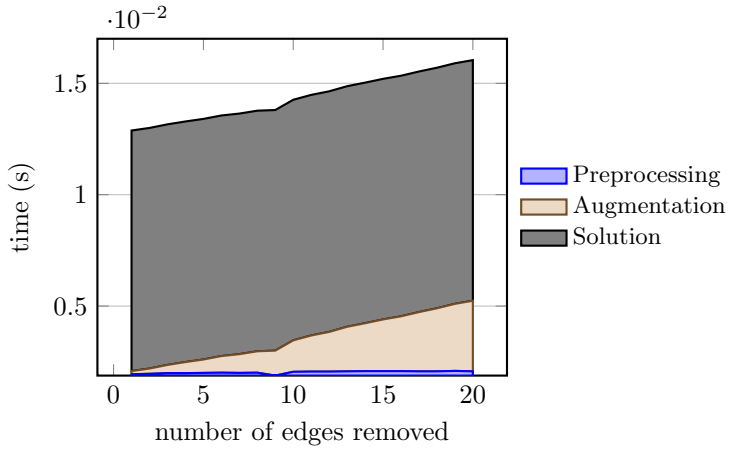


(a) Iterative method

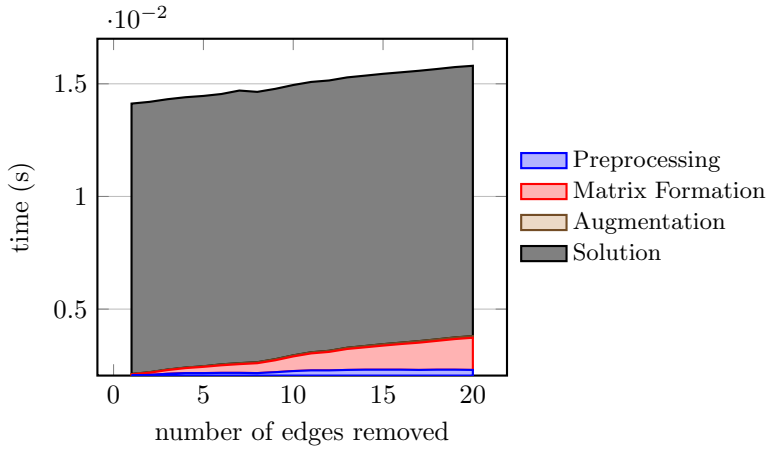


(b) Direct method

FIG. 3. *Breakdown of the time for the* 3,120*-bus Polish system*



(a) Iterative method



(b) Direct method

FIG. 4. *Breakdown of the time for the* 777,646*-bus generated system*

using PARDISO were 0.00735 seconds for the 3,120-bus Polish system, 0.039 seconds for the 14,070-bus WECC system, and 2.37 seconds for the 777,646-bus generated system. Although Oblio did not perform as well as PARDISO on the smaller problems, it provides the ability to extract the factors, which is essential for closure computation and the sparsity-exploiting triangular solves.

In Figure 1, we plot the time to compute the updated solution when up to 20 edges are removed from the grid. The augmented methods outperform both PARDISO and CHOLMOD on all three power grids. The time taken by PARDISO for the 777,646-bus generated system is not plotted in Figure 1c to better differentiate the relative performance of our methods with CHOLMOD. For this large grid, PARDISO took approximately 2.4 seconds for solving each modified system, which is two orders of magnitude ($149 - 186$ times) slower than our augmented iterative method. In comparison, CHOLMOD computed the solutions $1.47 - 5.09$ times slower than our augmented iterative method. We also observe that the augmented methods scale much better than CHOLMOD as the number of edges removed (size of the updates)

increases. However, the number of fill-ins, if any, introduced by CHOLMOD is insignificant, as we can see from Figure 2 that only the factor update time increases when the number of edges removed increases.

Figures 3 and 4 show the breakdown of the total time used in solving the updated systems using our augmented direct and iterative methods on the 3,120-bus Polish system and the 777,646-bus generated system. Here *Preprocessing* is the step of computing the closure of the modified rows and columns in the graph of $G(L)$, and extracting the necessary submatrix of L for solving for $\hat{\boldsymbol{x}}_3$ in (19) and (21). *Augmentation* refers to the step of solving (19) for the iterative method and (21) for direct method. *Matrix Formation* corresponds to the step of forming W as described in Subsection 3.3. *Solution* is the step of computing the solution to the modified system in (25).

It can be seen that for a small system the time is dominated by the augmentation part in (19) (Step 6 in Table 1) for the iterative method, or by the matrix formation of the reduced system in (21) (Step 3 in Table 1) in the direct method. Hence the product of m or t (the number of steps of the iterative solver) with $O(\rho)$ is the dominant term. On the other hand, for a large system the time is dominated by the computation of $\hat{\boldsymbol{x}}$ in (25) (Step 7 in Table 1), which has $O(|L|)$ time complexity.

The experimental results also indicate that the augmented solution methods do not lead to difficulties with solution accuracy. Table 2 summarizes the average relative residual norms for the solutions computed by our augmented methods, PARDISO and CHOLMOD.

5. Conclusions and future work. We have formulated two algorithms using an augmented matrix approach to solve linear systems of equations when the system is updated by a principal submatrix. The algorithms use either a direct method or a hybrid of direct and iterative methods. We applied the algorithms to assess the security of power grids, and demonstrated that we could do $N - k$ contingency analysis by removing $k = 20$ connections in a grid with 778,000 buses in about 16 milliseconds. The augmented solution methods have been experimentally shown to offer advantages in both speed and reliability, relative to a direct solver (two orders of magnitude faster), or a solver that updates the Cholesky factors (1.5 to 5 times faster), and scales better with an increasing k, the number of connections removed. We believe that our algorithms are able to solve much larger dynamic security analysis problems in the power grid than previous work.

In the future, we plan to extend our augmented solution method to problems where the updated system of equations has a different size than the original system, as in finite element applications [16].

Acknowledgments. We thank Dr. Jessica Crouch for collaborating with us in our earlier work on the augmented matrix approach to update solutions of linear systems of equations for visualizing and simulating surgery. We also thank Dr. Mallikarjuna Vallem for providing us the 777,646-bus generated system for our experiments on contingency analysis of power flow systems. We are grateful to two anonymous referees for their helpful comments, which have improved the presentation of our manuscript.

REFERENCES

[1] D. Bienstock, *Electrical Transmission System Cascades and Vulnerability*, Society for Industrial and Applied Mathematics, 2016, ch. 3, http://dx.doi.org/10.1137/1.9781611974164.

559 ch3.
560 [2] J. Bisschop and A. Meeraus, *Matrix augmentation and partitioning in the updating of the*
561 *basis inverse*, Mathematical Programming, 13 (1977), pp. 241–254, http://dx.doi.org/10.
562 1007/BF01584341.
563 [3] J. Bisschop and A. Meeraus, *Matrix augmentation and structure preservation in linearly*
564 *constrained control problems*, Mathematical Programming, 18 (1980), pp. 7–15, http://dx.
565 doi.org/10.1007/BF01588292.
566 [4] T. A. Davis and W. W. Hager, *Row modifications of a sparse Cholesky factorization*, SIAM
567 Journal on Matrix Analysis and Applications, 26 (2005), pp. 621–639, http://dx.doi.org/
568 10.1137/S089547980343641X.
569 [5] F. Dobrian and A. Pothen, *Oblio: Design and performance*, in Applied Parallel Computing.
570 State of the Art in Scientific Computing, J. Dongarra, K. Madsen, and J. Wasniewski,
571 eds., vol. 3732 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2006,
572 pp. 758–767, http://dx.doi.org/10.1007/11558958_92.
573 [6] J. Gilbert, *Predicting structure in sparse matrix computations*, SIAM J. Matrix Analysis and
574 Applications, 15 (1994), pp. 62–79, http://dx.doi.org/10.1137/S0895479887139455.
575 [7] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, *Sparse matrix methods in*
576 *optimization*, SIAM Journal on Scientific and Statistical Computing, 5 (1984), pp. 562–589,
577 http://dx.doi.org/10.1137/0905041.
578 [8] Intel Corporation, *Math Kernel Library Developer Reference*, 2015, https://software.intel.
579 com/en-us/articles/mkl-reference-manual.
580 [9] W. H. Kersting, *Radial distribution test feeders*, in Power Engineering Society Winter Meet-
581 ing, 2001. IEEE, vol. 2, 2001, pp. 908–912 vol.2, http://dx.doi.org/10.1109/PESW.2001.
582 916993.
583 [10] A. Kuzmin, M. Luisier, and O. Schenk, *Fast methods for computing selected elements of*
584 *the Green's function in massively parallel nanoelectronic device simulations*, in Euro-Par
585 2013 Parallel Processing, F. Wolf, B. Mohr, and D. Mey, eds., vol. 8097 of Lecture Notes
586 in Computer Science, Springer Berlin Heidelberg, 2013, pp. 533–544, http://dx.doi.org/10.
587 1007/978-3-642-40047-6_54.
588 [11] J. W. H. Liu, *The role of elimination trees in sparse factorization*, SIAM Journal on Matrix
589 Analysis and Applications, 11 (1990), pp. 134–172, http://dx.doi.org/10.1137/0611010.
590 [12] C. Maes, *A Regularized Active-set Method for Sparse Convex Quadratic Programming*,
591 PhD thesis, Stanford University, Nov. 2010, https://web.stanford.edu/group/SOL/
592 dissertations/maes-thesis.pdf.
593 [13] O. Schenk, M. Bollhöfer, and R. A. Römer, *On large-scale diagonalization techniques for*
594 *the Anderson model of localization*, SIAM Review, 50 (2008), pp. 91–112, http://dx.doi.
595 org/10.1137/070707002.
596 [14] O. Schenk, A. Wächter, and M. Hagemann, *Matching-based preprocessing algorithms to*
597 *the solution of saddle-point problems in large-scale nonconvex interior-point optimization*,
598 Computational Optimization and Applications, 36 (2007), pp. 321–341, http://dx.doi.org/
599 10.1007/s10589-006-9003-y.
600 [15] E. Wong, *Active-Set Methods for Quadratic Programming*, PhD thesis, University of Califor-
601 nia, San Diego, June 2011, http://ccom.ucsd.edu/~elwong/p/elw-thesis.pdf.
602 [16] Y.-H. Yeung, J. Crouch, and A. Pothen, *Interactively cutting and constraining vertices in*
603 *meshes using augmented matrices*, ACM Transactions on Graphics, 35 (2016), pp. 18:1–
604 18:17, http://dx.doi.org/10.1145/2856317.
605 [17] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, *MATPOWER: Steady-state*
606 *operations, planning, and analysis tools for power systems research and education*, IEEE
607 Transactions on Power Systems, 26 (2011), pp. 12–19, http://dx.doi.org/10.1109/TPWRS.
608 2010.2051168.