SIAM J. SCI. COMPUT.
Vol. 0, No. 0, pp. 000–000

EFFICIENT (PARTIAL) DETERMINATION OF DERIVATIVE MATRICES VIA AUTOMATIC DIFFERENTIATION*

WEI XU† AND THOMAS F. COLEMAN‡

Abstract. In many scientific computing applications involving nonlinear systems or methods of optimization, a sequence of Jacobian or Hessian matrices is required. Automatic differentiation (AD) technology can be used to accurately determine these matrices, and it is well known that if these matrices exhibit a sparsity pattern (for all iterates), then not only can AD take advantage of this sparsity for significant efficiency gains, AD can also determine the sparsity pattern itself, with some additional work in the first iteration. Practical nonlinear systems and optimization problems often exhibit patterns beyond just "zero-nonzero." For example, some elements may be duplicates of other elements at all iterates; some elements may be constant (not necessarily zero) for all iterates. Here we show how the popular graph-coloring approach to AD can be adapted to account for these cases as well, with resulting gains in efficiency. In addition, we address the problem of determining, by AD technology, a prescribed set of the entries of the Jacobian (or Hessian, in the optimization context) matrix.

Key words. Jacobian/Hessian matrix, matrix derivative computation, automatic differentiation, partial graph coloring

AMS subject classifications. 65K05, 65K10, 65H10, 90C30, 90C05, 68L10

DOI. 10.1137/11085061X

1. Introduction.

Computations in science and engineering often require the frequent determination of matrices of the first or second derivatives. For example, many methods for multidimensional zero-finding, or minimization of nonlinear functions, require a sequence of Jacobian matrices (first derivatives) or Hessian matrices (second derivatives), respectively. Determination of these matrices is often an expensive proposition; in fact, in many problems the dominant work in solving the entire problem is in the calculation of these derivative matrices. In light of this, many techniques have been developed to efficiently compute or approximate these matrices.

New tools and techniques have been developed in recent years that marry automatic differentiation (AD) technologies and the natural sparsity and structure that arises in large-scale problems, to allow for the efficient and exact determination of Jacobian and Hessian matrices [5, 6, 9]. In this paper we extend some of these sparse techniques to exploit the existence of constant nonzero terms, duplicate (nonconstant) entries in the Jacobian/Hessian matrices, and allow for partial determination of the derivative matrices. These are all very practical situations, and we illustrate that the sparse techniques developed in [3, 4, 5, 6] can be adapted to this setting to great advantage.

The general situation for repeated entries is the case where identical (nonconstant) copies of entries of the Jacobian/Hessian matrix occur. Since the entries are

*Submitted to the journal's Methods and Algorithms for Scientific Computing section October 7, 2011; accepted for publication (in revised form) February 25, 2013; published electronically DATE. This work was supported in part by the Ophelia Lazaridis University Research Chair (held by Thomas F. Coleman), the National Sciences and Engineering Research Council of Canada, and the Natural Science Foundation of China (project 11101310).

http://www.siam.org/journals/sisc/x-x/85061.html

†Department of Mathematics, Tongji University, Shanghai, China, 200092 (wdxu@tongji.edu.cn).

‡Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, Canada, N2L 3G1 (tfcoleman@uwaterloo.ca).

nonconstant they must be recomputed at every iterate x; however, since there are many "copies" only one "designate" of a set of copies needs to be computed by automatic differentiation. We explore how to do this in section 4.

Clearly some Hessian or Jacobian structures will not only have elements that are zero for all iterates x, but will also have some elements that are constant (but nonzero) for all iterates. Ignoring this fact can be costly not just because the same elements are recomputed at each iterate x, but their nonzero status can trigger additional unnecessary computations. For example, suppose the Jacobian structure of a nonlinear mapping $F(x) : \mathcal{R}^n \to \mathcal{R}^n$ is as follows, illustrated for the case $n = 5$:

$$J(x) = \begin{bmatrix} Y & X & X & X & X \\ & Y & & & \\ & & Y & & \\ & & & Y & \\ & & & & Y \end{bmatrix}, \tag{1.1}$$

where the nondiagonal nonzero entries, X, are (generally, different) constants and the diagonal entries are true *nonlinear* functions of the iterate x. A 1-sided automatic differentiator (i.e., "forward-mode") [10, 11] computes the product JV for an appropriate matrix V and then extracts the nonzero elements of J from this product. Because of the dense first row, matrix V must have at least n columns, and so the work to evaluate J at any argument x is proportional to $n \cdot \omega(F)$, where $\omega(F)$ is the work to evaluate F at any argument x. However, if the X-elements are constant (and known), then matrix V can be a single column vector (e.g., $V = (1, \ldots, 1)^T$), and it is clear that all the diagonal elements can then be determined from the product JV, with a simple arithmetic adjustment for the (1,1)-entry. So, given knowledge of the nonzero constants (in this case in the first row), the work to compute the Jacobian matrix J at an argument x by a 1-sided (forward mode) automatic differentiator is proportional to $\omega(F)$, i.e., the work required for a single evaluation of F. We generalize this example in section 5.

Some multidimensional zero-finding methods, as well as some nonlinear minimization techniques, request only a submatrix of the full derivative matrix at each iterate. For example, the Newton–Krylov method described in [13] requires only the lower triangular half of the sparse Jacobian matrix at each iterate. Therefore there is no need to determine the entire matrix. Consider here another example. Consider the minimization of a nonlinear function $f : \mathcal{R}^n \to \mathcal{R}^1$ via a linearized conjugate-gradient technique that uses the diagonal elements of the current Hessian matrix to form a preconditioner. It is clearly wasteful to compute the entire Hessian matrix, even if sparse, when only the diagonal elements are required.

To see this, suppose the Hessian matrix of f, i.e., matrix $H(x)$, is sparse for all x. As is the custom in this area, we consider a graph to present the structure.

A graph G is an ordered pair $G = (V,\ E)$, where V is a finite and nonempty set of vertices and E is a set of edges between two vertices. If $u \in V$, $v \in V$, and $(u, v) \in E$, then vertices u and v are said to be adjacent; otherwise they are nonadjacent. A path of length l in a graph is a sequence $v_1,\ v_2, \ldots, v_{l+1}$ of distinct vertices such that v_i is adjacent to v_{i+1} for $1 \le i \le l$. A p-coloring graph is defined on $\mathcal{G} = (V, E)$ if there exists a function ϕ defined as

$$\phi :\ V \to \{1, 2, \ldots, p\},$$

such that $\phi(u) \neq \phi(v)$ if $(u, v) \in E$. The smallest p for which $\mathcal{G}$ has a p-coloring is known as the chromatic number $\chi(\mathcal{G})$. A p-coloring ϕ of $\mathcal{G}$ partitions vertices V into

p groups $G_k,\ k = 1, 2, \ldots, p$, such that

$$G_k = \{u \in V :\ \phi(u) = k\}.$$

Let $G_A(H) = (V,\ E)$ be the corresponding adjacency graph of symmetric matrix H [5]. It is not hard to see that a p-coloring of $G_A(H)$ will yield an $n \times p$ matrix V from which the diagonal elements of H can be directly determined from the product HV, via the automatic differentiation. The work required is therefore proportional to $p \cdot \omega(f)$, where $\omega(f)$ is the work required to evaluate f at any argument x. However, if the entire sparse matrix H is determined via sparse AD, and then the diagonal terms extracted, the work required is proportional to $p_\pi \cdot \omega(f)$, where p_π is the number of colors required for a path coloring of $G_A(H)$; see [5]. Note that $\chi_\pi(G_A) \geq \chi(G_A)$ and, typically, $\chi_\pi(G_A) > \chi(G_A)$, where $\chi(G)$ is the chromatic number of graph G, and $\chi_\pi(G)$ is the path-chromatic number of a graph G (see [5]). There is no guarantee that graph-coloring heuristics produce p, p_π close to χ, χ_π, respectively, but for many practical problems the difference is not large [5]. This example class illustrates that when only a partial Jacobian (Hessian) matrix is required, it may be advantageous to tailor the AD tools with this in mind and avoid computing the entire matrix. We formalize and generalize these notions in section 2.

This paper is organized as follows. In section 2 we review the graph-theoretic approaches to the direct determination of a sparse Jacobian (Hessian) matrix by AD and then show how they can be modified for partial Jacobian (Hessian) determinations. In section 3 we consider substitution (indirect) partial determination of Jacobian (Hessian) matrices. In section 4 we discuss how AD can be adapted to the case where Jacobian/Hessian matrices have "copies." In section 5 we indicate how AD tools can be adapted to determine constant Jacobian (Hessian) terms. Two applications of the partial matrix determination technique are suggested in section 6, and results of computational experiments are given in section 7. Finally, in section 8 we summarize our results and discuss future directions of inquiry.

2. (Partial) direct determination of Jacobian and Hessian matrices. Generally, automatic differentiation (AD) methods compute matrices that represent the product of a matrix of derivatives with a given input matrix. Such products are produced accurately without first computing the matrix of derivatives. *Direct determination* of the underlying matrix of derivatives, i.e., the Jacobian (or Hessian) matrix, is a method that chooses this input matrix so that once the product matrix is computed, the nonzero elements of the Jacobian (or Hessian) can be extracted from the product by a simple diagonal system solver.

2.1. Direct determination of (partial) Jacobian matrices via automatic differentiation. First we consider 1-sided (forward-mode) direct determination. Let NZ be the set of nonzero (i, j)-elements of matrix J, for all x, and let $Y \subseteq NZ$ be the subset of elements to be computed (via AD) at the current point. The complement of Y is denoted by $N : N = NZ - Y$. We are interested in efficiently determining the values of the Jacobian matrix J, at current iterate x, corresponding to set Y. We note that it is not possible to simply ignore the set N, i.e., treat elements in N as zero elements, since their nonzero values may conflict with terms with indices in Y. The intersection graph to be colored, for the forward-mode application of AD, is given by the following definition.

DEFINITION 2.1. *Assuming set Y is not empty, then a Y-intersection graph $\mathcal{G}_I^Y(J) = (V,\ E_I^Y)$ with $i \in V$ and $j \in V$ has an edge in the set of edges E_I^Y, i.e.,*

$(i,j) \in E_I^Y$ *if there exists* k *such that* either $(k,i) \in Y$ *and* $(k,j) \in NZ$ or $(k,i) \in NZ$ *and* $(k,j) \in Y$.

Note that if $Y = NZ$, then we are reduced to the usual sparsity condition: the column intersection graph $\mathcal{G}_I(J) = (V,\ E_I)$ is defined where n vertices $V = \{v_1, \ldots, v_n\}$ correspond to the n columns of J with $(i,j) \in E_I$ if there exists an index k such that $(k,i) \in NZ$ and $(k,j) \in NZ$. Clearly $E_I^Y \subseteq E_I$ and therefore $\chi(\mathcal{G}_I^Y(J)) \leq \chi(\mathcal{G}_I(J))$, where $\chi(\mathcal{G})$ is the chromatic number of graph G. We illustrate these concepts with a simple Jacobian structure J and set Y below. For example, given the arrow matrix J in (2.1), where X represents a nonzero entry, if Y is equal to the set of all nonzeros of J, NZ, then set $V = I_{n\times n}$ to compute the whole J, where $I_{n\times n}$ is the $n \times n$ identity matrix; if Y represents all nonzeros of the lower triangular part of J, then $V = (e_1,\ e_2 + e_3 + e_4 + e_5)$, where e_i is the ith column of $I_{n\times n}$.

$$J = \begin{bmatrix} X & X & X & X & X \\ X & X & & & \\ X & & X & & \\ X & & & X & \\ X & & & & X \end{bmatrix}. \tag{2.1}$$

Clearly, only two vectors are needed to compute the lower triangular part of J, while n vectors are required to compute all nonzeros of J.

The best we can do in the 1-sided case when $Y = NZ$ is to find $p \geq \chi(\mathcal{G}_I^Y(J))$; for many practical problems good graph-coloring heuristics come close to this bound. The product JV can be computed by the 1-sided application of AD, i.e., "forward-mode" AD when $Y \subseteq NZ$.

In [5, 6] a superior 2-sided approach is developed: the matrices W, V are determined by a "bicoloring" technique. If F is a vector-valued differentiable function, $F : \mathcal{R}^n \to \mathcal{R}^m$, with (sparse) Jacobian matrix J, then given a source code to evaluate F at any argument $x \in \mathcal{R}^n$, an AD tool can, in general, determine the pair of products $(W^T J,\ JV)$, where W is a matrix with m rows and V is a matrix with n rows. If the nonzero elements of J can be directly extracted from this pair (by solving an implicit diagonal system), then we say that J is *directly* determined. The amount of work required is proportional to $(p+q)\cdot\omega(F)$, where p is the number of columns of matrix W, q is the number of columns of matrix V, and $\omega(F)$ is a measure of the work to evaluate F at an arbitrary argument x. The challenge is to choose matrices V, W such that the sum $p+q$ is as small as possible subject to the unique and efficient direct determination of J from the AD-computed pair $(W^T J,\ JV)$. We note that some AD packages only allow for a "forward-mode" computation, which means that only the second member of the pair, JV, is computed. Methods for determining appropriate pairs V, W (or just the singleton in the forward-mode case) with a small value of $p+q$ (or just q in the forward-mode case) are given in [3].

Here we explore how to extend this technique to the case where only a designated subset of the nonzero elements, $Y \subseteq NZ$, is required. Again we note that it is not sufficient just to treat the elements in $NZ - Y$ as designated zeros since the AD computations will then produce unresolved conflicts. The first step in [5] is to permute rows and columns of J to produce a jagged partition (see [5]), as in Figure 1. In Figure 1, the matrix J is permuted to $\tilde{J}$ via $\tilde{J} = PJQ = [J_C,\ J_R]$, where P, Q are permutation matrices chosen to induce the two sparse intersection graphs $G_I(J_C)$ and $G_I(J_R^T)$. J_C is the part to be colored by columns, while J_R is to be colored by rows. Our approach here is similar to the 1-sided approach introduced above,

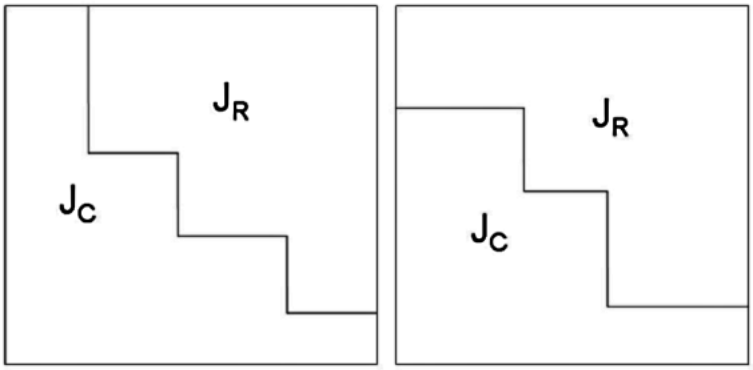

FIG. 1. *Possible partitions of the matrix* $\tilde{J} = P \cdot J \cdot Q$.

bearing in mind we care only to compute the elements in Y. The first step is to find a good permutation (i.e., permutation matrices P, Q) so that the subsequent bicoloring step is expected to be bounded by small chromatic numbers. Here we use algorithm MNCO provided in [5, sect. 4.3]. Assume that J is partitioned (as illustrated in Figure 1). The general approach now is to associate a suitable intersection graph with each partition J_C, J_R and then color the corresponding graphs with as few colors as possible to induce "thin" matrices V, W. We define the partial intersection graph for J_C, $\mathcal{G}_I(J_C) = (V_C,\ E_C)$ as follows.

DEFINITION 2.2. *Given a subset* Y *of the set of all nonzeros of* J, NZ, *the partial intersection graph* $\mathcal{G}_I^Y(J_C) = (V_C, E_C)$ *can be defined as follows:*

1. *vertex* $j \in V_C$ *if* $nnz(\text{column } j \cap J_C \cap Y) \neq 0$, *where* $nnz(x)$ *refers to the number of nonzeros in a vector or matrix* x;
2. $(r, s) \in E_C$ *if* $r \in V_C$, $s \in V_C$, *and there exists* k *such that* either $J_{kr} \neq 0$ *and* $(k, s) \in Y \cap J_C$ or $J_{ks} \neq 0$ *and* $(k, r) \in J_C \cap Y$.

Similarly, we define the partial intersection graph for J_R, $\mathcal{G}_I^Y(J_R) = (V_R, E_R)$.

DEFINITION 2.3. *Given a subset* Y *of the set of all nonzeros of* J, NZ, *the partial intersection graph* $\mathcal{G}_I^Y(J_R) = (V_R, E_R)$ *can be defined as follows:*

1. *vertex* $j \in V_R$ *if* $nnz(\text{row } j \cap J_R \cap Y) \neq 0$;
2. $(r, s) \in E_R$ *if* $r \in V_R$, $s \in V_R$, *there exists* k *such that* either $J_{rk} \neq 0$ *and* $(s, k) \in J_R \cap Y$ or $J_{sk} \neq 0$ *and* $(r, k) \in J_R \cap Y$.

Valid (p, q)-colorings of the two intersection graphs $\mathcal{G}_b(J_R)$ and $\mathcal{G}_b(J_C)$, respectively, will yield matrices W and V (of dimensions $m \times p$ and $n \times q$, respectively) such that the elements of Y can be determined directly from the AD-computed pair $(W^T J,\ JV)$.

2.2. Direct determination of (partial) Hessian matrices via automatic differentiation. Direct determination of the set of nonzeros, NZ, of a symmetric matrix H can be viewed as a (restricted) graph-coloring problem. In particular, assuming that all the diagonal elements of H belong to NZ, then in [5] it is proven that a *path p-coloring* of the vertices of the adjacency graph of H, $G_A(H) = (V,\ E_A)$, yields a matrix $V \in \mathcal{R}^{n \times p}$ such that the nonzero elements of H can be directly determined from the elements of the product HV (this product can be determined by requesting that an AD tool twice differentiate nonlinear function $f : \mathcal{R}^n \to \mathcal{R}^1$) along the columns of matrix V. (In fact this result is essentially a *characterization* of

direct determination of a symmetric matrix if we restrict the matrix V to a matrix that induces a partitioning of the columns of H.)

A *path p-coloring* of the vertices of a graph is a coloring using p colors such that every path of at least 3 edges uses at least 3 colors. In [5], it is shown how to add edges to $G_A(H)$ to produce a graph $\tilde{G}_A(H) = (V,\ \tilde{E}_A)$, $E_A \subseteq \tilde{E}_A$, such that a *p-coloring* of $\tilde{G}_A(H)$ is a *path p-coloring* of $G_A(H)$. The procedure advocated in [5] is to first symmetrically permute the rows and columns of H (with the aim of adding few edges to produce $\tilde{G}_A(H)$; see [5]), and then apply the following definition.

DEFINITION 2.4. *The intersection graph $\mathcal{G}_I(H) = (V, \tilde{E}_A)$ based on the symmetric matrix H can be defined as follows: for each nonzero $h_{ij}(i \le j)$ of H, $(i,j) \in \tilde{E}_A$ if there exists $k \ge i$ such that $H_{k,i} \ne 0$ and $H_{k,j} \ne 0$.*

We now modify Definition 2.4 to get a definition of partial estimation of H on a subset Y.

DEFINITION 2.5. *Given a subset Y of the set of all nonzeros of H, NZ, the partial intersection graph $\mathcal{G}_I^Y(H) = (V, \tilde{E}_A)$ can be defined as follows: for each index (i,j) of Y with nonzero $h_{ij}(i \le j)$, $(i,j) \in \tilde{E}_A$ if there exists $k \ge i$ such that $(k,i) \in Y$ and $H_{k,j} \ne 0$.*

For example, consider the case where H is a 8×8 symmetric band matrix of full bandwidth 5 as shown in (2.2), where X represents a nonzero entry,

$$H = \begin{bmatrix} X & X & X & & & & & \\ X & X & X & X & & & & \\ X & X & X & X & X & & & \\ & X & X & X & X & X & & \\ & & X & X & X & X & X & \\ & & & X & X & X & X & X \\ & & & & X & X & X & X \\ & & & & & X & X & X \end{bmatrix}. \tag{2.2}$$

Suppose that Y is the set of diagonal elements. In this case, we only need three vectors, e.g., $V = (e_1 + e_4 + e_7,\ e_2 + e_5 + e_8,\ e_3 + e_6)$, to obtain the diagonal elements directly. On the other hand, five vectors are needed to directly estimate the whole matrix, e.g., $V = (e_1 + e_4 + e_7,\ e_2 + e_5 + e_8,\ e_3 + e_6, e_4,\ e_5)$.

3. (Partial) indirect determination of Jacobian (and Hessian) matrices. Direct methods can be improved upon, in terms of work required to determine a Jacobian matrix, by allowing for a *substitution* process to determine the nonzero elements of the Jacobian [5]. Specifically, in [5] it was shown that it is often possible to efficiently determine the sparse Jacobian matrix J by determining the pair $(W^T J,\ JV)$ by using forward and reverse modes of AD, and then recovering the nonzero elements of J via a substitution process. In [5], the matrices W, V are determined by a "bicoloring" technique. Here we explore how to extend this technique to the case where only a designated subset of the nonzero elements, $Y \subseteq NZ$, is required. Again we note that it is not sufficient just to treat the elements in $N = NZ - Y$ as designated zeros since the AD computations will produce unresolved conflicts.

Both direct and substitution processes locate matrices V and W to compute all nonzero elements of J. The difference between a direct and a substitution process is that the direct process extracts all nonzero elements from the pair $(W^T J,\ JV)$ by solving an implicit diagonal linear system, while the substitution process extracts nonzeros by solving a triangular system. In [5], it is illustrated that the substitution process often requires fewer matrix-vector products than the direct process does.

There are two steps for the substitution process. First, permute and partition the structure of J: $\tilde{J} = P \cdot J \cdot Q = [J_C|J_R]$ as in Figure 1. Then define *appropriate* intersection graphs $\mathcal{G}_C$, $\mathcal{G}_R$ based on the partition $[J_C|J_R]$. The difference between the direct and substitution process here is the definition of the intersection graphs $\mathcal{G}_C$ and $\mathcal{G}_R$ and the procedure to extract nonzeros of J from pair $(W^TJ,\ JV)$. The following definitions show how to construct the intersection graphs $\mathcal{G}_C$ and $\mathcal{G}_R$ for J_C and J_R, respectively, with the restriction on the subset Y.

DEFINITION 3.1. *Given a subset Y of the set of all nonzeros of J, NZ, the partial intersection graph $\mathcal{G}_I^Y(J_C) = (V_C,\ E_C)$ for substitution determination can be defined as follows:*

1. *for each column j of J, if $nnz(\text{column } j \cap Y \cap J_C) \neq 0$, then $j \in V_C$;*
2. *for $i \in V_C$, $j \in V_C$, $(i,j) \in E_C$ if there exists k such that* either $(k,i) \in Y \cap J_C$ *and* $(k,j) \in NZ \cap J_C$ or $(k,i) \in Y \cap J_C$ *and* $(k,j) \in J_R \cap N$.

In the substitution determination, if entries $a_C \in J_C \bigcap Y$ and $a_R \in J_R \bigcap Y$ belong to the same set, then we can determine a_R by row coloring first, then determine a_C by a single substitution step. On the other hand, if a_C and $\tilde{a}_R \in J_R \bigcap N$ belong to the same set, we cannot determine a_C from a single substitute process since $\tilde{a}_R$ is not determined by the row coloring. As a result, a_C and $\tilde{a}_R$ have to be separated into two different sets. Similarly, we can get the substitute determination for $\mathcal{G}_I^Y(J_R) = (V_R,\ E_R)$.

DEFINITION 3.2. *Given a subset Y of the set of all nonzeros of J, NZ, the partial intersection graph $\mathcal{G}_I^Y(J_R) = (V_R,\ E_R)$ for substitution determination can be defined as follows:*

1. *for each row j of J, if $nnz(\text{row } j \cap Y \cap J_R) \neq 0$, then $j \in V_R$;*
2. *for $i \in V_R$, $j \in V_R$, $(i,j) \in E_R$ if there exists k such that* either $(i,k) \in Y \cap J_R$ *and* $(j,k) \in NZ \cap J_R$ or $(i,k) \in Y \cap J_R$ *and* $(j,k) \in J_C \cap N$.

For example, we have a sparse Jacobian matrix J as in (3.1),

$$J = \left[\begin{array}{ccccc} J_{11} & & J_{13} & J_{14} & \\ & & J_{23} & & \\ J_{31} & J_{32} & & & J_{35} \\ & J_{42} & & J_{44} & \\ & J_{52} & J_{53} & & \end{array}\right], \tag{3.1}$$

where J_{ij} is the (i,j)th element of J. Let $Y = \{J_{ij} \mid i \geq j \text{ and } J_{ij} \neq 0\}$, i.e., the lower triangular part of J. The matrix J is partitioned into J_C, the left lower part in (3.1), and J_R, the right upper part in (3.1). Then Figures 2 and 3 are the intersection graphs of J_C and J_R restricted to Y via the direct and substitution estimation, respectively.

From Figure 2, the coloring of $\mathcal{G}_c$ and $\mathcal{G}_R$ leads to the following matrices V and W and results in the computation of JV and W^TJ, where entry X means it is not included in set Y, so we do not need to determine its value:

$$V = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad JV = \begin{bmatrix} J_{11} & 0 & X \\ 0 & 0 & X \\ J_{31} & X & 0 \\ 0 & X & 0 \\ 0 & J_{52} & J_{53} \end{bmatrix},$$

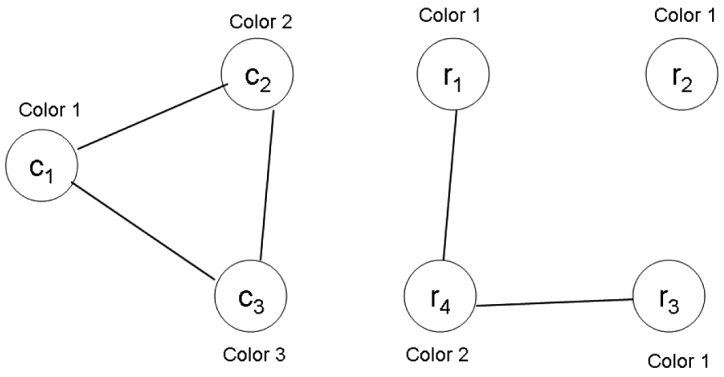

FIG. 2. *Graphs $\mathcal{G}_I(J_C)$ and $\mathcal{G}_I(J_R)$ for the direct process.*

$$W = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} W^T J = \begin{bmatrix} 0 & J_{42} & 0 & J_{44} & 0 \\ X & J_{32} & X & X & X \end{bmatrix}.$$

From Figure 3, the resulting matrices V and W and the computation JV and $W^T J$ are as follows:

$$V = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \qquad JV = \begin{bmatrix} J_{11} & X \\ 0 & X \\ J_{31} + J_{32} & 0 \\ X & 0 \\ J_{52} & J_{53} \end{bmatrix},$$

$$W = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} W^T J = \begin{bmatrix} 0 & J_{42} & 0 & J_{44} & 0 \\ X & J_{32} & X & X & X \end{bmatrix}.$$

Thus, all nonzeros in Y can be determined easily from the pair $(W^T J,\ JV)$.

When the matrix is symmetric, for example, the Hessian matrix H, then the intersection graph can be modified as the following definition restricted to subset $Y \subseteq NZ$.

DEFINITION 3.3. *Given a subset Y of the set of all nonzeros of H, NZ, the partial intersection graph $\mathcal{G}_I^Y(H) = (V, \tilde{E}_A)$ can be defined as for each index (i,j) of Y with nonzero $h_{ij} (i < j)$, $(i,j) \in \tilde{E}_A$ if there exists $k \geq i$ such that* either *$k < j$ and $(k,i) \in Y$ and $(k,j) \in N$* or *$k \geq j$ and $(k,i) \in Y$ and $(k,j) \in NZ$.*

4. Automatic differentiation and repeated Jacobian/Hessian entries. Symmetry is one obvious situation where entries in the (symmetric) Jacobian matrix (often a Hessian matrix) are repeated. That is, for all (i,j), $J_{ij}(x) = J_{ji}(x)$. AD techniques have been developed to use this symmetry knowledge and significantly decrease the work required to compute $J(x)$, especially in the case where $J(x)$ is also sparse. For example, consider the Jacobian matrix J, corresponding to the arrow

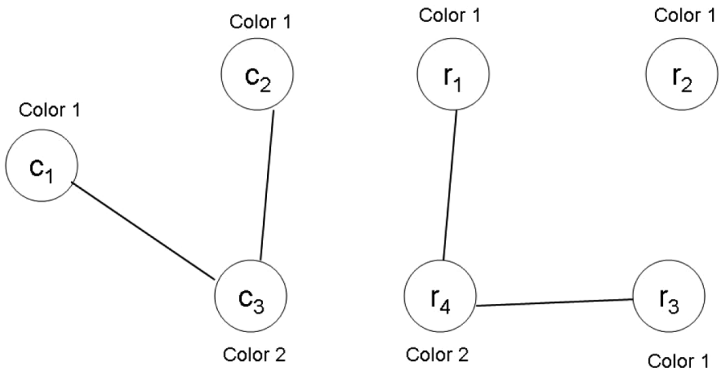

FIG. 3. *Graphs $\mathcal{G}_I(J_C)$ and $\mathcal{G}_I(J_R)$ for the substitution process.*

function, in (2.1). If J is symmetric, then for the 1-sided coloring using the product $J \cdot V$, the matrix V can be chosen $V = (e_1,\ e_2 + e_3 + e_4 + e_5)$. Otherwise, ignoring symmetry, the matrix V has to be $n \times n$.

Here we consider the more general situation where the pattern of repetition is more complex than symmetry and the repetition can be more than pairwise. A given Jacobian entry may be repeated several times in a haphazard pattern (the repetition and pattern is constant for all x). Clearly it can be inefficient to require the calculation of all copies of a Jacobian entry: in principle it is possible to require the computation of just one of the copies, say a *designated* member of the set of copies.

By sampling the Jacobian (or Hessian) matrix at several points on initiation, an AD tool can determine the sets of equal-valued entries. Assume there are M disjoint equivalence sets, $I_1, \ldots, I_M$, each of cardinality at least two: if $(i, j) \in I_k$, $(r, s) \in I_k$, then $J_{ij}(x) = J_{rs}(x)$ for all x. Clearly it is necessary that only one member of each equivalence set be explicitly computed by AD (and the other members of the equivalence set can then be assigned that computed value). Our strategy then is to require that just one member of each equivalence set be computed by AD; we call that member the *designate* of that set.

There are two questions to address. (1) How is the designate for each set $I_1, \ldots, I_M$, chosen? (2) How should the AD/coloring approach be modified? First we focus our attention on the first problem: choosing the equivalence set designates.

The main idea behind each of the cases discussed below is to attempt to choose designates that will result in removing as many edges as possible, in a locally greedy fashion, from the associated intersection graph.

4.1. 1-sided estimation of Jacobian matrices (in the presence of duplicates). The basic idea behind our proposed algorithm is to successively consider all equivalence sets that have yet to be assigned a designate. Order the indices included in these remaining equivalence sets according to the number of nonzeros in all the rows touched by the members of the corresponding equivalence set. The member with the highest score is then selected to be the designate for its equivalence set, and that set is then removed from further consideration (and all the other members of that set are "marked" as *copies*.). For $r = 1, \ldots, m$, let NZ_r be the number of nonzeros in row r; then our proposed algorithm can be formalized as follows.

ALGORITHM 1. *Given a Jacobian matrix $J \in R^{m\times n}$ and all its repeated nonzero elements divided into M disjoint sets $I_1, I_2, \ldots, I_M$. The following algorithm determines the designate of each equivalence set.*

% ASSIGN a score to each member of each equivalence set.
for *each $(i,j) \in I_k$, $(k = 1, \ldots, M)$, and for each $r \neq i$ with $(r,s) \in I_k$ for some s score$(i,j) = \sum NZ_r$,*
end
% CHOOSE the designates.
Let $S = I_1 \bigcup I_2 \bigcup \cdots \bigcup I_M$, with $D = \emptyset$;
while *S is not empty*
Find (i_*, j_*) such that score$(i_*, j_*) \geq$ score(i,j) $\forall (i,j) \in S$;
Let t be such that $(i_*, j_*) \in I_t$;
Index (i_*, j_*) is assigned to be the designate for set I_t;
Set $D = D \cup \{(i_*, j_*)\}$; ord$(t) = i_*$;
Mark all other members of I_t as copies;
$S \leftarrow S - \{I_t\}$;
end

Thus, the intersection graph for determination of the Jacobian matrix J considering the repeat entries can now be constructed as indicated by the following definition.

DEFINITION 4.1. *Let $D \subseteq NZ$ be the set of designates and singletons in NZ, and let C be the set of all copies, that is, all those indices identified by $I_1, \ldots, I_M$ with the exception of the designated set D. The intersection graph $\mathcal{G}_I^D(J) = (V, E_I^D)$ can be defined as follows:*

1. *vertex $j \in V$ if j is a column of J;*
2. *$(i,j) \in E_I^D$ if $i \in V$, $j \in V$, and there exists k such that* either *$(k,j) \in D$ and $(k,i) \in D$* or *$(k,i) \in D \cap I_s$ and $(k,j) \in I_t$ with ord$(t) \geq$ ord(s).*

4.2. 2-sided estimation of Jacobian matrices (in the presence of duplicates). The challenge in adapting the ideas presented above to the substitution approach is that the substitution process imposes a certain partial ordering. In principle this may conflict with a designate selection process. For example, consider Figure 1. The substitution process requires that nonzero elements in Block jj be resolved in order that elements in Block kk be determined when $kk > jj$. Therefore the scoring process (above) must be adjusted to ensure that the designate for each equivalence set is in the lowest possible numbered section (with reference to Figure 1). Our approach is to first determine the jagged partition

$$\tilde{J} = PJQ = [J_C,\ J_R] \tag{4.1}$$

using Algorithm MNCO in [5, sect. 4.3]. This method determines the partition (4.1) as well as the block ordering $B_1, \ldots, B_s$ indicating the partial order in which the elements of NZ must be resolved (see Figure 4).

The following algorithm to assign scores to the members of each equivalence set I_k $(k = 1, \ldots, M)$ is similar to Algorithm 1, with the major exception that only duplicates in the lowest numbered block for each equivalence set are considered for designate selection. Let $NZ_{*,s}$ be the number of nonzeros in column s of J_R if $(r,s) \in J_R$, and let NZ_r be the number of nonzeros in row r in J_C if $(r,s) \in J_C$. Then the algorithm can be formalized as follows.

ALGORITHM 2. *Given a Jacobian matrix $J \in R^{m\times n}$ and its all nonzero elements divided into M disjoint sets $I_1, I_2, \ldots, I_M$. The following algorithm determines the designate of each equivalence set:*

% ASSIGN a score to each member of each equivalence set.
Let $LB(k)$ *be the lowest numbered index such that*
$I_k \cap B_{LB(k)} \neq \emptyset$ $(k = 1, 2, \ldots, M)$*;*
for *each* $(i, j) \in I_k$, $(k = 1, \ldots, M)$
if $(i, j) \notin B_{LB(k)}$
$score(i, j) = 0$*;*
elseif *There exists* r, *s and* $r \neq i$, *s.t.* $(r, s) \in I_k$
$score(i, j) = \sum NZ_r$*;*
else *There exists* r, *s and* $s \neq j$, *s.t.* $(r, s) \in I_k$
$score(i, j) = \sum NZ_{*,s}$*;*
end
end
% CHOOSE the designates.
Let $S = I_1 \bigcup I_2 \bigcup \cdots \bigcup I_M\}$, *with* $D = \emptyset$*;*
while S *is not empty*
Find (i_*, j_*) *such that* $score(i_*, j_*) \geq score(i, j)$ $\forall (i, j) \in S$*;*
Let t *be such that* $(i_*, j_*) \in I_t$*;*
Index (i_*, j_*) *is assigned to be the designate for set* I_t*;*
Set $D = D \cup \{(i_*, j_*)\}$*;* $ord(t) = i_*$*;*
Mark all other members of I_t *as copies;*
$S \leftarrow S - \{I_t\}$*;*
end

The intersection graph for determination of the Jacobian matrix J by substitution can now be constructed based on the following definition.

DEFINITION 4.2. *Let* $D \subseteq NZ$ *be the set of designates and singletons in* NZ, *and let* C *be the set of all copies, that is, all those indices identified by* $I_1, \ldots, I_M$, *with the exception of the designated set* D. *The intersection graph* $\mathcal{G}_I^Y(J_C) = (V_C^Y,\ E_C^Y)$ *satisfies the following:*

1. *for each column* l *of* J, $l \in V_C^Y$ *if* $nnz(Y \cap$ *column* $l \cap J_C) \neq 0$*;*
2. *for* $i \in V_C^Y$, $j \in V_C^Y$, $(i, j) \in E_C^Y$ *if there exists* k *such that* either $(k, i) \in D \cap J_C$ *and* $(k, j) \in D \cap J_C$ or $(k, i) \in D \cap I_s \cap J_c$ *and* $(k, j) \in I_t \cap J_C$ *with* $ord(t) \geq ord(s)$.

Note that the corresponding row graph, $\mathcal{G}_I^D(J_R) = (V, E_R^D)$, can be constructed in an analogous manner.

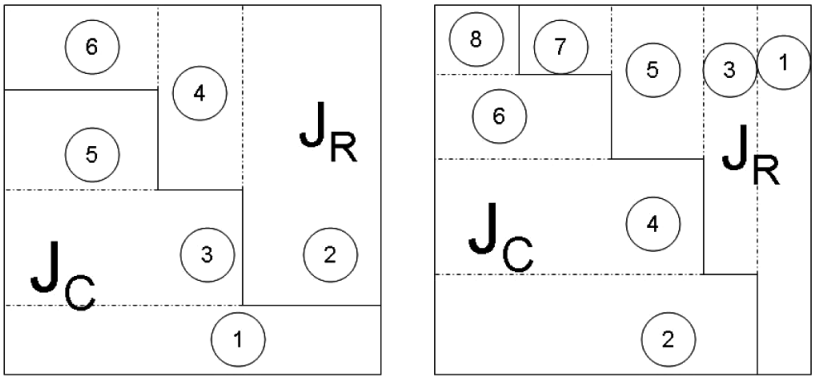

FIG. 4. *Graphs for substitution ordering.*

5. Automatic differentiation and constant terms in Jacobian and Hessian matrices. In this section we consider that, in some applications, the corresponding Jacobian/Hessian matrix has constant nonzero elements, which is a special case of the repeated entries discussed in the previous section. For example, consider the Broyden function in [1]. The explicit form of the Broyden function $F^B: R^n \to R^n$ is

$$F_i^B = (3 - 2x_i)x_i - x_{i-1} - 2x_{i+1} + 1,$$

where $x_0 = x_{n+1} = 0$ and $F^B = [F_1^B, F_2^B, \ldots, F_n^B]^T$. Then, given a vector x, its corresponding Jacobian matrix is

$$J = \begin{bmatrix} 3-2x_1 & -2 & & & & \\ -1 & 3-2x_2 & -2 & & & \\ & -1 & 3-2x_3 & -2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 3-2x_n \end{bmatrix}. \tag{5.1}$$

It is obvious that J is a tridiagonal matrix with constant off-diagonal elements. If we have knowledge of the constant elements in J, then only the main diagonal need be computed to construct J at iterate x. Thus, matrix $V = (e_1+e_3+e_5+e_7,\ e_2+e_4+e_6)$ can yield J from the product $J \cdot V$ when $n = 7$, given knowledge of the constant off-diagonal terms. The total cost to construct J is $2 \cdot \omega(F^B)$. However, if we ignore the fact that there are two nonzero constants in J, that is, we compute every nonzero in J, then matrix V needs three columns, e.g., $V = (e_1 + e_4 + e_7,\ e_2 + e_5,\ e_3 + e_6)$. The total cost is $3 \cdot \omega(F^B)$.

Another example is the linear function–full rank problem in [1]. As shown in [1], the explicit form of this function $F^L: R^n \to R^m$ is

$$F_i^L = x_i - \frac{2}{m}\left(\sum_{j=1}^{n} x_j\right) - 1, \qquad 1 \le i \le n,$$

$$F_i^L = -\frac{2}{m}\left(\sum_{j=1}^{n} x_j\right) - 1, \qquad n+1 \le i \le m,$$

where $F^L = [F_1^L,\ F_2^L,\ \ldots, F_m^L]^T$. Obviously, the Jacobian matrix J has the form

$$J = \begin{bmatrix} X & C & C & \cdots & C \\ C & X & C\cdots & C & \\ \vdots & & & & \vdots \\ C & C & \cdots & C & X \\ C & C & \cdots & & C \\ \vdots & & & & \vdots \\ C & C & \cdots & & C \end{bmatrix},$$

where $C = -\frac{2}{m}$ and $X = 1 - \frac{2}{m}$. If we do not consider the constant elements of J, then V needs n columns, e.g., $V = I$, to recover J via the forward-mode AD. However, if we have knowledge of the constants in J, then only one vector e_1 is required to construct matrix J. Typically, the sparsity of matrix J has to be computed in the first iteration before applying AD.

Once the sets of constant Jacobian/Hessian entries have been detected on initialization (or on input), and values stored, the technique for handling these nonzeros in the procedure for applying AD to determine the other (varying) nonzeros is almost identical to our procedure presented in section 4 for handling duplicates. There is one exception: in this case clearly there is no need for determining "designates" since all copies are known from the outset. In all other aspects the procedure is the same.

Once the constant entries in the Jacobian/Hessian matrix are detected at the first iteration, they can be stored as one of the sparsity computational components since these entries have no need to be determined again for the next iterations. To update the Jacobian/Hessian matrix, we just need to employ the partial determination technique or repeated entries technique proposed in previous sections to compute all nonconstant entries only and keep the constant entries unchanged. Therefore, this updating strategy will save a lot of computational time when frequent Jacobian or Hessian updating is required—the Newton method, for example.

6. Applications. One application of the partial Jacobian estimation is to construct the preconditioner for the Krylov subspace method to solve nonlinear equations [13]. In [13], Xu and Coleman proposed a Jacobian-free Newton method, which did not require the evaluation of the whole Jacobian matrix, but required only a lower triangular or diagonal estimation [7] of the Jacobian for the preconditioner construction. Then they employed the forward-mode AD and the Krylov subspace method to solve the nonlinear equations. The advantage of this method is that it is Jacobian free, but produces accurate Jacobian-vector product estimation. As shown in [13], the number of function valuations is reduced significantly compared to the Newton method, with full Jacobian evaluations in some cases based on the direct partial Jacobian evaluation.

Another application of the partial Jacobian estimation is to solve the power flow problem with the quasi-Newton method [12]. First, the Jacobian matrix for each Newton method is separated into the constant part and the updated part. The constant part means some rows of the Jacobian matrix are kept constant for a number of iterations to reduce computational cost. If the whole Jacobian matrix is kept constant, the cost is lowest but the method converges linearly. Thus, Semlyen and de León proposed a partial Jacobian update in the solution strategy. They updated some rows of the Jacobian to accelerate the convergence while keeping the other rows constant. In other words, Newton steps were combined with constant Jacobian steps and partial Jacobian updates to get an efficient quasi-Newton method. Then, with the partial updated Jacobian matrix, the matrix modification lemma was used to update its corresponding LU decomposition. Thus, only the backward and forward substitutions were required to solve the Jacobian system. However, Semlyen and de León did not specify any methodology to compute the updated Jacobian rows given a power flow function. The finite difference method is an easy implementation for the Jacobian approximation, but it can only approximate the Jacobian vector product Jv. As we show in section 1 in [13], the finite difference method is not suitable to approximate rows of the Jacobian matrix. When one row is dense, the cost to approximate the row is similar to the cost to approximate the whole Jacobian matrix. Therefore, with the proposed methodology in sections 2 and 3, the partial Jacobian rows can be updated efficiently and accurately with AD.

7. Computational experiments. In this section, we will present some numerical experiments on the direct/indirect partial bicoloring method and the constant structure idea in section 4. All experiments are carried on a workstation with an

TABLE 1
Testing matrices collection.

Matrix	Type	Size	Nonzeros
garon1	Power network	3175×3175	84723
circuit_1	Circuit simulation problem	2624×2624	35823
rajat13	Circuit simulation problem	7598×7598	48762
nopoly	Undirected weighted graph	10774×10774	70842
cond_mat-2003	Undirected weighted graph	31163×31163	240058
TF15	Combinatorial problem	6334×7742	80057
g7jac080sc	Subsequent theoretical problem	9506×9506	394808
orani678	Economic problem	2529×2529	90158
jan99jac020sc	Economic problem	6774×6774	33744
jan99jac040	Economic problem	13694×13694	72734
g7jac050sc	Economic problem	14760×14760	145157
mark3jac080	Economic problem	36609×36609	214643
mark3jac100	Economic problem	45769×45769	268563
fp	Computational fluid problem	22560×22560	1014951
g7jac140	Computational fluid problem	14822×14822	7158404
ted_A	Thermal problem	10605 ×10605	424587
Si10H16	Quantum chemistry problem	17077×17077	875923
lp_wood1p	Linear programming	244×2595	70126
fomell	Linear programming	12142×24460	71264
sctap1-2b	Linear programming problem sequence	15390×33858	99454
FA	Directed graph problem	10617×10617	72176
EAT_SR	Directed weighted problem	23219×23219	325589
cage10	Directed weighted graph	11397×11397	150645
C-39	Optimization problem	9271×9271	116578
C-43	Optimization problem	11125×11125	123659
C-42	Optimization problem	10471×10471	110285
C-48	Optimization problem	18354×18354	166080
onetone2	Frequency-domain circulate simulation	36057×36057	222596
zeros_nopss_13k	Power system	13296×13296	48827
bips98_1450	Power system	11305×11305	44678
aug3dcgq	Extended system-3D PDE	35543×35543	128115

AMD Phenom II X6 1090T 3.2GHz processor, 16 GB RAM, and 1 TB hard drive running 64-bit Windows 7 and MATLAB R2009a. As mentioned in previous sections, the AD technique is employed to compute the Jacobian-vector products Jv and $w^T J$. Here, we use the MATLAB AD package ADMAT 2.0 [2] for the implementation of the forward- and reverse-mode AD.

All testing sparse matrices are from the Tim Davis collection [8] and its subcollections. All selected testing matrices are listed in Table 1. The first column of Table 1 lists the names of the test matrices. The second column is the application background from which selected matrices were generated. The third and fourth columns are the matrix sizes and numbers of nonzeros of tested matrices, respectively. In our experiments, we do not consider the 1-sided coloring determination since it has already been illustrated in [5, 13] that the bicoloring determination is much more efficient than the 1-sided coloring whether through full matrix determination [5] or lower triangular partial determination [13].

First, we compare the effectiveness of the direct/indirect partial bicoloring de-

I've edited this sentence slightly, based on your reply to my query. Please check that it makes sense to you.

TABLE 2
Number of matrix-vector products for full bicoloring estimation, direct partial bicoloring, and indirect partial coloring to estimate the lower triangular portion and diagonal of sparse matrices.

Matrix	Full matrix	Lower triangular		Diagonal	
	Direct	Direct	Substitute	Direct	Substitute
markjac080	37	33	25	13	10
g7jac080sc	150	111	108	49	47
g7jac140	169	128	121	58	55
garon1	51	45	45	9	8
circuit_1	139	125	124	68	68
rajat13	28	19	16	9	9
nopoly	15	13	13	6	6
orani678	245	200	200	112	110
jan99jac020sc	68	44	44	13	13
jan99jac040	75	61	61	17	17
zeros_nopss_13k	32	30	17	6	4
mark3jac100	37	33	24	13	10
fp	424	265	263	19	19
ted_A	178	178	123	20	20
zeros_nopss_13k	31	31	18	5	4
FA	53	42	42	3	3
aug3dcgp	31	31	17	6	4
C-39	206	111	111	4	4
C-43	111	83	64	4	4
C-42	67	46	34	6	5
C-48	96	45	45	6	5
EAT_SR	264	195	195	14	14
cond_mat-2003	113	91	89	2	2
onetone2	141	100	100	49	49

termination with the bicoloring direct full determination in [5]. All results are listed in Table 2. The names of the test matrices are listed in the first column. The second column is the number of matrix-vector products for full bicoloring determination. The third and fourth columns are the number of matrix-vector products using the direct and indirect partial bicoloring method to determine the lower triangular portion of the tested matrices, respectively. The fifth and sixth columns are the number of matrix-vector products using the direct and indirect partial bicoloring method to determine the diagonal of matrices, respectively. These results illustrate that the partial determination can reduce the number of matrix-vector products significantly, while the indirect method requires even fewer matrix-vector products than the direct method does in some cases.

Next, we test the constant structure idea introduced in section 4 on some large sparse matrices. In this experiment, we will determine the sparse matrices by the bicoloring method with and without the constant structure idea. Results are listed in Table 3. The first column of Table 3 is the name of the tested matrices. The second column is the number of matrix-vector products required by the full bicoloring method in [5]. The third column is the number of matrix-vector products by the bicoloring method taking into consideration the constant structure as in section 4. The last column shows the number of distinct nonzero entries in the tested matrices. Our results illustrate that more repeated entries result in fewer matrix-vector products.

TABLE 3
Number of matrix-vector products for the full bicoloring estimation with/without constant structure idea.

Matrix	Full bicoloring	Constant structure bicoloring	Number of distinct nonzeros
garon1	51	47	11677
circuit_1	130	75	10676
rajat13	28	20	6940
nopoly	15	3	12
TF15	49	5	15
jan99jac020sc	68	41	4955
jan99jac040	75	47	8968
g7jac050sc	141	141	145157
zeros_nopss_13k	31	30	20297
mark3jac100	37	31	127503
fp	424	373	319677
ted_A	178	126	130214
Si10H16	506	257	26630
lp_wood1p	107	33	3327
FA	53	11	1002
sctap1-2b	23	8	34
aug3dcgq	11	11	24915
cage10	66	5	237
C-39	206	187	33849
C-43	111	63	36081
C-42	67	59	51711
C-48	96	78	71512
bips98_1450	31	31	19959
EAT_SR	264	8	132
cond_mat-2003	113	40	3063
onetone2	51	20	14671

Results in Table 3 indicate that the constant structure idea is an effective way to reduce the cost for the sparse Jacobian/Hessian matrix evaluation, especially when the Jacobian/Hessian matrix has to be evaluated many times. The overhead of detecting constants is about $O(nnz)$, where nnz is the number of nonzeros in the detected matrix. It needs to go through all nonzero entries and separate them into sets. If the Jacobian/Hessian matrix is computed for only one or two times, it is not economic to employ this detecting idea. However, significant savings will result when these derivative matrices are evaluated many times, such as in the Newton method for nonlinear problems.

8. Concluding remarks. The accurate and efficient determination of derivatives, especially gradients and Jacobian and Hessian matrices, is a fundamental computational task in scientific computing. Efficiencies gained here can affect countless computing tasks in scientific modeling and engineering. The field of automatic differentiation (AD) is generally making it easier and faster to compute these derivative quantities, and with great accuracy.

Significant work has been done (and subsequent methods are now used in applications) with respect to tailoring AD techniques to exploit (and determine) sparsity in Jacobian and Hessian matrices. Indeed, in some cases the resulting savings in

computing time can be several orders of magnitude.

In this paper the observation is made that additional structure/intelligence may be available, and this information can further increase AD efficiency in a significant way. Specifically, duplication in the matrix entries—and the special case of constant values—can be determined (in a randomized manner), and this intelligence can be used to further increase the efficiency of AD. In this paper, graph-coloring techniques, used to solve the sparsity problem, are adapted to the case of duplicate matrix values (and constants). The experimental results illustrate that there can be a significant decrease in computing time if such structure is detected in this fashion.

In addition, the observation is made that in some numerical approaches to scientific computing problems, only a partial Jacobian (or, sometimes, partial Hessian) matrix is required. We show how to adapt the graph-based sparsity techniques to this case, with subsequent additional gains in efficiency.

Further extensions of this work are possible. For example, while effective techniques have been developed in this paper, there certainly is no optimality proof, and there is scope for the development of alternative techniques. In addition we have focused on the case where there are duplicate entries in the Jacobian (and Hessian) matrix. Duplicate entries may be known at the outset, or determined by an initial randomized process. However, a generalization of this idea is to determine "almost equivalent" sets, that is, sets such that if one, or several, members are determined, then all members of the set are easily determined.

A simple example of this is when all the members of the set satisfy a simple linear relationship. Such a generalization could potentially lead to significant increases in efficiencies, but there are many algorithmic questions to be explored.

9. Appendix: Bipartite graph interpretation. A graph $\mathcal{G}_b = ([V_1\ V_2],\ E)$ is bipartite if the vertex set V is partitioned into two disjoint sets V_1 and V_2 such that every edge in E connects a vertex from V_1 to a vertex from V_2. Given a matrix $A \in R^{m\times n}$, we can construct a bipartite graph $\mathcal{G}_b(A) = ([V_1,\ V_2],\ E)$, where $V_1 = [r_1, r_2, \ldots, r_m]$, $V_2 = [c_1, c_2, \ldots, c_n]$, and c_j represents the jth column of A, and r_i represents the ith row of A. Then if a_{ij} is nonzero, there is an edge between r_i and c_j, i.e., $(r_i, c_j) \in E$.

For the 1-sided AD computation, i.e., compute the product JV via forward-mode AD, define the partial bipartite path p-coloring of $\mathcal{G}_b$ on V_2, where $V_2 = [c_1, c_2, \ldots, c_n]$ and c_i is the ith column of J, as follows.

DEFINITION 9.1. *Let $\mathcal{G}_b^F = ([V_1,\ V_2],\ E)$ be a bipartite graph where $F \subseteq E$. A mapping $\phi : V_2 \to \{0, 1, 2, \ldots, p\}$ is a partial bipartite path p-coloring of $\mathcal{G}_b^F$ on V_2 restricted to F if the following conditions hold:*

1. *If $i \in V_1,\ j \in V_2$, and $(i, j) \in F$, then $\phi(j) \neq 0$.*
2. *If $i \in V_1,\ j \in V_2,\ k \in V_2$, $(i, k) \in E$, and $(i, j) \in F$, then $\phi(j) \neq \phi(k)$.*

The following theorem establishes when a bipartition is consistent with direct determination with a portion of matrix J, given the nonzero subset Y.

THEOREM 9.1. *Let A be an $m \times n$ matrix with corresponding partial bipartite graph $\mathcal{G}_b^Y(A) = ([V_1,\ V_2],\ E)$. The mapping $\phi : [V_1, V_2] \to \{0, 1, \ldots, p\}$ induces a bipartition $[G_R, G_C]$ with $|G_R| + |G_C| = p$, consistent with direct determination if and only if ϕ is a partial bipartite path p-coloring of $\mathcal{G}_b^Y(A)$.*

Proof. ($\Leftarrow$) Assume ϕ is a partial bipartite path p-coloring of $\mathcal{G}_b^F(A)$ on V_2 restricted to F. According to Definition 9.1, we have, for $a_{ij} \in Y,\ \phi(j) = \alpha \neq 0$. That is, column j belongs to the group $\mathcal{G}_{\phi(j)} = \{c_j \mid \phi(j) = \alpha\}$. Assume there exists an element in Y that cannot be determined directly, that is, $a_{ij} \in Y,\ a_{ik} \in NZ$, and

$\phi(j) = \phi(k)$. This contradicts condition 2 in Definition 9.1.

($\Rightarrow$) Assume that ϕ induces a partial bipartite path p-coloring of $\mathcal{G}_b$ on V_2 restricted to F. It is obvious that condition 1 in Definition 9.1 is satisfied. We will construct condition 2. Assume that there are indices $i \in V_1,\ j \in V_2,\ k \in V_2,\ (i,j) \in F,\ (i,k) \in E$. According to condition 1, $\phi(j)$ and $\phi(k)$ are positive. If $\phi(j) = \phi(k)$, then the element $a_{ij} \in F$ cannot be determined directly, which contradicts the assumption. Thus, we have $\phi(j) \neq \phi(k)$. □

A path p-coloring of a graph can be defined to be a vertex coloring using p colors with the property that every path of at least three edges uses at least three colors [3, 4]. Here, we give a new definition of a partial bipartite path p-coloring of a graph with the modification in [5], that is, we assign a color "0" to some vertices, i.e., $\phi(x) = 0$, which means that vertex i is not assigned a color.

DEFINITION 9.2. *Assume we are given a subset Y of E and a bipartite graph $\mathcal{G}_b^Y = ([V_1,\ V_2],\ E)$. A mapping $\phi : [V_1, V_2] \to \{0, 1, 2, \ldots, p\}$ is a bipartite path p-coloring of $\mathcal{G}_b^Y$ if the following conditions hold:*

1. *Adjacent vertices have different assignments, that is, for $i \in V_1,\ j \in V_2$, if $(i,j) \in Y$, then $\phi(i) \neq \phi(j)$; or if $(i,j) \notin Y$ and there exists $k \in V_2$ such that $(i,k) \in Y$, then $\phi(i) \neq \phi(j)$; or if $(i,j) \notin Y$ and there exists $k \in V_1$ such that $(k,j) \in Y$, then $\phi(i) \neq \phi(j)$.*
2. *The set of positive colors used by vertices in V_1 is disjoint from the set of positive colors used by vertices in V_2, i.e., $i \in V_1,\ j \in V_2 \ \Rightarrow\ \{\phi(i) \neq \phi(j)$ or $\phi(i) = \phi(j) = 0\}$.*
3. *Vertices i and j are adjacent to vertex k, with $\phi(k) = 0$. If $i, j \in V_2$ and $(k,i) \in Y$ or $(k,j) \in Y$, then $\phi(i) \neq \phi(j)$; if $i, j \in V_1$ and $(i,k) \in Y$ or $(j,k) \in Y$, then $\phi(i) \neq \phi(j)$.*
4. *For a path with three edges, say $r_i - c_j - r_k - c_l$ if $(k,j) \in Y$, then at least three colors are used for this edge.*

The following theorem establishes that a valid p-coloring of $\mathcal{G}_b^Y(J)$ yields an $n \times p$ matrix V such that the product JV will lead to direct determination of the required nonzero elements $Y \subseteq NZ$.

THEOREM 9.2. *Given a subset Y of all nonzeros NZ of an $m \times n$ matrix A, let A have a corresponding partial graph $\mathcal{G}_b^F(A) = ([V_1,\ V_2],\ E)$, where if $a_{ij} \in Y$, then $(r_i, c_j) \in F$. The mapping $\phi : V_2 \to \{0, 1, \ldots, p\}$ consistent with the direct determination of Y if and only if ϕ is a partial bipartite path p-coloring of $\mathcal{G}_b^F(A)$ on V_2 restricted to F.*

Proof. The proof is similar to that for Theorem 3.1 in [5].

($\Leftarrow$) Assume that ϕ is a partial bipartite path p-coloring of $\mathcal{G}_b(A)$, including a partial bipartite $(G_R,\ G_C)$ of rows and columns of A. If this partial bipartition is not consistent with direct determination, there exists a nonzero entry $a_{i,j}$ of A that does not satisfy the conditions in Definition 9.2. This can happen only if one of the following situations holds.

1. $\phi(i) = 0$, $\phi(j) \neq 0$, $a_{ij} \in Y$, and there exists a column q with $a_{iq} \neq 0$, such that $\phi(j) = \phi(q)$, which leads to a contradiction of condition 3 of Definition 9.2.
2. $\phi(i) = 0$, $\phi(j) \neq 0$, $a_{ij} \notin Y$, and there exists a column q such that $a_{iq} \in Y$ with $a_{iq} \neq 0$ such that $\phi(j) = \phi(q)$. This contradicts with condition 3 of Definition 9.2.
3. $\phi(i) \neq 0$, $\phi(j) = 0$, $a_{ij} \in Y$, and there exists a row q with $a_{qj} \neq 0$ such that $\phi(i) = \phi(q)$, which contradicts condition 3.
4. $\phi(i) \neq 0$, $\phi(j) = 0$, $a_{ij} \notin Y$, and there exits a row q such that $a_{qj} \in Y$ with

"three colors" correct? (You're correction stated "...then at least colors are used", and I assumed that was a typo.)

Please check that your corrections to change \phi(c_j) to \phi(j), etc., throughout the proof were done correctly.

$a_{qj} \neq 0$ such that $\phi(i) = \phi(q)$, which contradicts condition 3.

5. $\phi(i) \neq 0$, $\phi(j) \neq 0$, $a_{ij} \in Y$, and there exists a column q and a row p such that columns j and q are in the same group with $a_{iq} \neq 0$ and rows i and p are in the same group with $a_{pj} \neq 0$. This implies that ϕ is a 2-coloring of the path $(r_p - c_j - r_i - c_q)$, which contradicts condition 4.
6. $\phi(i) \neq 0$, $\phi(j) \neq 0$, $a_{ij} \notin Y$, and there exists a column p and a row q such that rows i and p are in the same group with $a_{pj} \neq 0$ and $a_{pj} \in Y$, and columns j and q are in the same group with $a_{iq} \neq 0$. Then it implies that the path $(r_i - c_j - r_p - c_q)$ is colored by two colors, which contradicts condition 4.

($\Rightarrow$) Assume that ϕ induces a partial bipartite consistent with direct determination of A. It is obvious that condition 1–3 are satisfied. It remains to establish condition 4. Assume there is a bicolored path $r_i - c_j - r_k - c_l$, where $(r_k,\ c_j) \in Y$, $\phi(i) = \phi(k)$, and $\phi(j) = \phi(l)$. It is clear from condition 3 that two colors are positive. If $(r_k,\ c_j) \in Y$, then the element a_{kj} cannot be determined directly since $\phi(i) = \phi(k)$ and $\phi(j) = \phi(l)$. ∎

REFERENCES

[1] B. M. Averick, R. G. Carter, and J. J. Moré, *The MINIPACK-2 Test Problem Collection*, Working paper, Technical Memorandum 150, Argonne National Laboratory, 1991.

[2] Cayuga Research Associates, LLC, *ADMAT-2.0 User's Guide*, www.cayugaresearch.com, 2009.

[3] T. F. Coleman and J.-Y. Cai, *The cyclic coloring problem and estimation of sparse Hessian matrices*, SIAM J. Algebraic Discrete Methods, 7 (1986), pp. 221–235.

[4] T. F. Coleman and J. J. Moré, *Estimation of sparse Hessian matrices and graph coloring problems*, Math. Programming, 28 (1984), pp. 243–270.

[5] T. F. Coleman and A. Verma, *The efficient computation of sparse Jacobian matrices using automatic differentiation*, SIAM J. Sci. Comput., 19 (1998), pp. 1210–1233.

[6] T. F. Coleman and A. Verma, *ADMIT-1: Automatic differentiation and MATLAB interface toolbox*, ACM Trans. Math. Software, 26 (2000), pp. 150–175.

[7] J. Cullum and M. Tůma, *Matrix-free preconditioning using partial matrix estimation*, BIT, 46 (2006), pp. 711–729.

[8] T. Davis and Y. Hu, *The university of Florida sparse matrix collection*, ACM Trans. Math. Software, 38 (2011), 1.

[9] A. H. Grebremedhin, F. Manne, and A. Pothen, *What color is your Jacobian? Graph coloring for computing derivatives*, SIAM Rev., 47 (2005), pp. 629–705.

[10] A. Griewank and G. F. Corliss, eds., *Automatic Differentiation of Algorithms: Theory, Implementation, and Applications*, SIAM, Philadelphia, PA, 1991.

[11] A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, 2nd ed., SIAM, Philadelphia, PA, 2008.

[12] A. Semlyen and F. de León, *Quasi-Newton power flow using partial Jacobian updates*, IEEE Trans. Power Systems, 16 (2001), pp. 332–339.

[13] W. Xu and T. F. Coleman, *Solving Nonlinear Equations with Newton-Krylov Method Based on Automatic Differentiation*, Working Paper, 2011.