The efficient computation of sparse Jacobian matrices using automatic differentiation *

Thomas F. Coleman[†] Arun Verma[‡]

November 22, 1995

Abstract

This paper is concerned with the efficient computation of sparse Jacobian matrices of nonlinear vector maps using automatic differentiation (AD). Specifically, we propose the use of a graph coloring technique, bi-coloring, to exploit the sparsity of the Jacobian matrix J and thereby allow for the efficient determination of J using AD software. We analyze both a direct scheme and a substitution process. We discuss the results of numerical experiments indicating significant practical potential of this approach.

Key words. sparse Jacobian matrices, nonlinear systems of equations, nonlinear least squares, graph coloring, bi-coloring, automatic differentiation, computational differentiation, sparse finite differencing, partition problem, NP-complete problems, ADOL-C

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

1 Introduction

The efficient numerical solution of nonlinear systems of algebraic equations, $F(x) : \Re^n \to \Re^m$, usually requires the repeated calculation or estimation of the matrix of first derivatives, the Jacobian matrix, $J(x) \in \Re^{m \times n}$. In large-scale problems matrix J is often sparse and it is important to exploit this fact in order to efficiently determine, or estimate, matrix J at a given argument x. This paper is concerned with the efficient calculation of sparse Jacobian matrices by the judicious application of automatic differentiation techniques. Specifically, we show how to define "thin" matrices V and W such that the nonzero elements of J can easily be extracted from the calculated pair $(W^T J, JV)$. Given an arbitrary n-by-t_V matrix V, product JV can be directly calculated using automatic differentiation in the "forward mode"; given an arbitrary m-by-t_W matrix W, the product $W^T J$ can be calculated using automatic differentiation in the "reverse mode", e.g., [11, 13].

*This research was partially supported by the Applied Mathematical Sciences Research Program (KC-04-02) of the Office of Energy Research of the U.S. Department of Energy under grant DE-FG02-90ER25013, and in part by the Advanced Computing Research Institute, a unit of the Cornell Theory Center which receives major funding from the National Science Foundation and IBM Corporation, with additional support from New York State and members of its Corporate Research Institute.

†Computer Science Department and Center for Applied Mathematics, Cornell University, Ithaca NY 14850.

‡Computer Science Department, Cornell University, Ithaca NY 14850.

The forward mode of automatic differentiation allows for the computation of product JV in time proportional to $t_V \cdot \omega(F)$ where $\omega(F)$ is the time required to evaluate F. This fact leads to the following practical question. Given the structure of a sparse Jacobian matrix J, how can a matrix V be chosen so that the nonzeros of J can easily be determined from the product JV? A good solution is offerred by the sparse finite-differencing literature [4, 5, 6, 7, 8, 10] and adapted to the automatic differentiation setting [1]. Partition the columns of J into a set of groups G_C, where the number of groups in G_C is denoted by $|G_C|$, such that the columns in each group $G \in G_C$ are *structurally orthogonal*[1]. Each group $G \in G_C$ defines a column v of V: $v_i = 1$ if and only if column i is in group G; otherwise, $v_i = 0$. It is clear that the nonzeros of J can be immediately "identified" from the computed product JV. Graph coloring techniques, applied to the column intersection graph of J, can be used to try and produce a partition G_C with low cardinality $|G_C|$. This, in turn, induces a thin matrix V, i.e., construct $V \in \Re^{n \times t_V}$ where $t_V = |G_C|$. However, it is not always possible to ensure that $|G_C|$ is small: consider a sparse matrix J with a single dense row.

Alternatively, the reverse mode of automatic differentiation allows for the computation of the product $W^T J$ in time proportional to $t_W \cdot \omega(F)$ where $\omega(F)$ is the time required to evaluate F, and $W \in \Re^{m \times t_W}$. The "transpose" of the argument above can lead to an efficient way to determine J. That is, apply graph coloring techniques to the row intersection graph of J to induce a thin matrix W; compute $W^T J$ via the reverse mode of automatic differentiation (takes time proportional to $t_W \cdot \omega(F)$); trivially extract the nonzeros of J from the computed matrix $W^T J$. Of course it is easy to construct examples where defining a thin matrix W is not possible – consider the case where J has a dense column.

Clearly there are problems where a row-oriented approach is preferable, there are problems where a column-oriented approach is better. Unfortunately, it is easy to devise problems where neither approach is satisfactory: let J have both a dense row and a dense column. This is exactly when it may pay to use both modes of automatic differentiation simultaneously: compute a pair $(W^T J, JV)$, for suitable choices of W and V, and extract the nonzero elements of J from this computed pair of thin matrices.

Our concern in this paper is with efficiency with respect to number of floating point operations or $flops$. We do not concern ourselves with space requirements in this study. However, it should be noted that the reverse mode of automatic differentiation often requires significantly more space than the forward mode: if space is tight then our suggested approach, which involves application of both forward and reverse modes, may not be possible. There is current research activity on reducing the space requirements of the reverse mode of automatic differentiation, e.g., [12].

We note that an independent proposal regarding sparse Jacobian calculation is made by Hossain and Steihaug [15]: a graph-theoretic interpretation of the direct determination problem is given and an algorithm based on this interpretation is provided. In this paper we proffer a new direct method and we also propose a substitution method, both based directly on the Jacobian structure. We compare our direct and substitution methods, numerically, and we discuss the round-off properties of the substitution method. In addition, we interface our graph coloring software to the automatic differentiator ADOL-C, [14], and report on a few preliminary computational results.

The remainder of the paper is organized as follows. In §2, we review the relevant aspects of automatic differentiation, both forward and reverse modes. In §3, we formalize the combinatorial problems to be solved both from a matrix point of view and in terms of graph theory. We propose both a direct determination problem and a substitution problem. In §4, we propose "bi-coloring"

[1] Two nonzero n-vectors v, w are *structurally orthogonal* if $v_i * w_i = 0, \ i = 1 : n$.

approaches to both the direct determination and "determination by substitution" problems. The bi-coloring technique produces matrices V and W where JV is subsequently determined via the forward mode of automatic differentiation, $W^T J$ is detemined via the reverse mode. Typically, the column dimensions of V and W will be small: the cost of the application of automatic differentiation is proportional to the sum of the column dimensions of V and W (times the work to evaluate F).

In §5 we present and discuss various numerical experiments. The experiments indicate that our bi-coloring approach can significantly reduce the cost of determining J (over one-sided Jacobian determination).

The substitution method we propose consistently outperforms the direct method. However, the substitution calculation increases the chance of round-off contamination. This effect is discussed in §6. We end the paper in §7 with some concluding remarks and observations on possible directions for future research. Specifically, we note that while sparsity is a symptom of underlying structure in a nonlinear problem, it is not a necessary symptom. Moreover, it is often possible to exploit structure in the absence of sparsity and apply AD tools "surgically" to efficiently obtain the Jacobian matrix J.

2 Basics of automatic differentiation

Automatic differentiation is a chain rule based technique for evaluating the derivatives analytically (and hence without any truncation errors) with respect to input variables of functions defined by a high-level language computer program. In this section we briefly review the basics of automatic differentiation, borrowing heavily from [11, 13].

A program computing the function $z = F(x), F : \Re^n \to \Re^m$, can be viewed as a sequence of scalar assignments $v_i = \psi < v_j >_{j \to i}$, where the vector v can be thought of as set of ordered variables such that $v_i, i \to j$, is computed before v_j using the set of variables $\{v_k | k \to i\}$. Here ψ_j represent elementary functions, which can be arithmetic operations and/or univariate transcedental functions.

Ordering the variables as above, we can partition variables v_j into three vectors.

$$\begin{aligned} x &\equiv (v_1, v_2, \ldots, v_n) \text{ (independent)} \\ y &\equiv (v_{n+1}, v_{n+2}, \ldots, v_{n+p}) \text{ (intermediate)} \\ z &\equiv (v_{n+p+1}, v_{n+p+2}, \ldots, v_{n+p+m}) \text{ (dependent)} \end{aligned}$$

In general, the number of intermediate variables is much larger than the dimensions of the problem, i.e., $p \gg m, n$.

Assume that all these elementary functions ψ_i are well defined and have continuous elementary partials $c_{ij} = \frac{\partial \psi_i}{\partial v_j}, j < i$. Assuming with out loss of generality that the dependent variables y do not themselves occur as arguments of elementary functions, we can combine the partials c_{ij} into the $(p+m) \times (n+p)$ matrix

$$C = \left(c_{n+i,j}\right)_{1 \le i \le p+m}^{1 \le j \le n+p}$$

Unless elementary functions with more than two arguments are included in the library, each row of C contains either one or two non zero entries. We define a number

$$q \equiv nnz(C) \le 2p \tag{2.1}$$

Also, since work involved in an elementary function is proportional to number of arguments, it follows that

$$\omega(F) = \sum_{i=1}^{p+m} \omega(\psi_{n+i}) \sim q \tag{2.2}$$

where $\sim$ indicates proportionality. Because of the ordering relation the square matrix C is upper trapezoidal with $n-1$ nontrivial superdiagonals. Thus C can be partitioned as

$$C = \left(\begin{array}{c|c} & \\ A & L \\ & \\ \hline B & M \end{array}\right) \tag{2.3}$$

where

$$A \in \Re^{p\times n}, B \in \Re^{m\times n}, L \in \Re^{p\times p}, M \in \Re^{m\times p},$$

and L is strictly lower triangular. Application of the chain rule yields:

$$\begin{pmatrix} A & L-I \\ B & M \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} 0 \\ -z \end{pmatrix}. \tag{2.4}$$

If we eliminate the intermediate vector Δy from above (2.4), we get an expression(the Schur complement) for the Jacobian:

$$J \equiv B - M(L-I)^{-1}A = B + M(I-L)^{-1}A \tag{2.5}$$

Since $I - L$ is a unit lower triangular matrix, the calculation of the matrix products $\tilde{A} \equiv (I-L)^{-1}A$ and $\tilde{M} \equiv M(I-L)^{-1}$ leads to two natural ways to compute J:

$$J = B + M\tilde{A} \quad or \quad J = B + \tilde{M}A. \tag{2.6}$$

The alternative expressions for J given in (2.6) define the two basic *modes* of automatic differentiation, *forward* and *reverse*.

The *forward mode* corresponds to computing the rows of $\tilde{A}$, one by one, as the corresponding rows of $[A, L-I]$ are obtained from successive evaluation of elementary functions. Since this amounts to solutions of n linear systems with lower-triangular matrix $[I-L]$, followed by multiplication of dense columns of $\tilde{A}$ by M, the total computational effort is roughly $n \cdot q$ or $n \cdot \omega(F)$.

The *reverse mode* corresponds to computing $\tilde{M}$ as solution to linear system $(I-L)^T\tilde{M}^T = M^T$. This back-substitution process can begin only after all elementary functions and their partial derivatives have been evaluated. Since this amounts to the solution of m linear systems with lower triangular matrix $[I-L]$, followed by multiplication of dense rows of $\tilde{M}$ by A, total computational effort is roughly $m \cdot q$ or $m \cdot \omega(F)$.

We are interested in computing products of the form JV and W^TJ. Product JV can be computed:

$$JV \quad = \quad BV + M[(I-L)^{-1}(AV)]$$

which can clearly be done in time proportional to $t_V \cdot \omega(F)$ when $V \in \Re^{n \times t_V}$. Analagously, product $W^T J$ can be computed:

$$W^T J \;\; = \;\; W^T B + [(W^T M)(I - L)^{-1}]A$$

which can be done in time proportional to $t_W \cdot \omega(F)$ assuming $W \in \Re^{m \times t_W}$.

An important subcase worthy of special attention is when F is a scalar function, i.e., $m = 1$. In this case the Jacobian matrix corresponds to the transpose of the gradient of F and is a single row vector. Note that the complexity arguments applied to this case imply that the reverse mode of AD yields the gradient in time proportional to $\omega(F)$ whereas the forward mode costs $n \cdot \omega(F)$. The efficiency of the forward mode evaluation of the gradient can be dramatically increased - i.e., the dependence on n is removed - if F has structure that can be exploited [2].

3 Partition problems and graph theory

Our basic task is to efficiently determine thin matrices V, W so that the nonzero elements of J can be readily extracted from the information $(W^T J, JV)$. The pair of matrices $(W^T J, JV)$ is obtained from the application of both modes of automatic differentiation: matrix $W^T J$ is computed by the reverse mode, the forward mode determines JV. The purpose of this section is to more rigorously formulate the question of determining suitable matrices V, W, first in the language of "partitions", and then using graph theoretic concepts.

We begin with an example illustrating the usefulness of simultaneously applying both modes of AD, forward and reverse. Consider the following n-by-n Jacobian, symmetric in structure but not in value:

$$J \; = \; \begin{pmatrix} \square & \triangle & \triangle & \triangle & \triangle \\ \square & \Diamond & & & \\ \square & & \Diamond & & \\ \square & & & \Diamond & \\ \square & & & & \Diamond \end{pmatrix}. \tag{3.1}$$

It is clear that a partition of columns consistent with the direct determination of J requires n groups. This is because a "consistent column partition" requires that each group contain columns that are structurally orthogonal and the presence of a dense row implies each group consists of exactly one column. Therefore, if matrix V corresponds to a "consistent column partition" then V has n columns and the work to evaluate JV by the forward mode of AD is proportional to $n \cdot \omega(F)$. By a similar argument, and the fact that a column of J is dense, a "consistent row partition" requires n groups. Therefore, if matrix W corresponds to a "consistent row partition" then W has n rows and the work to evaluate $W^T J$ by the reverse mode of AD is proportional to $n \cdot \omega(F)$.

Definition 3.1 *A* **bi-partition** *of a matrix A is a pair (G_R, G_C) where G_R is a row partition of a* **subset** *of the rows of A , G_C is a column partition of a* **subset** *of the columns of A.*

In this example the use of a bi-partition dramatically decreases the amount of work required to determine J. Specifically, the total amount of work required is proportional to $3 \cdot \omega(F)$. To see this define $V = (e_1, e_2 + e_3 + e_4 + e_5)$; $W = (e_1)$, where we follow the usual convention of representing the i^{th} column of the identity matrix with e_i. Clearly elements $\square, \Diamond$ are directly determined from the product JV; elements $\triangle$ are directly determined from the product $W^T J$.

The basic idea is to partition the rows into a set of groups G_R and the columns into a set of groups G_C, with $|G_R| + |G_C|$ as small as possible, such that every nonzero element of J can be directly determined from *either* a group in G_R or a group in G_C.

Definition 3.2 *A bi-partition (G_R, G_C) of a matrix A is* **consistent with direct determination** *if for every nonzero a_{ij} of A, either column j is in a group of G_C which has no other column having a nonzero in row i, or row i is in a group of G_R which has no other rows having a nonzero in column j.*

Clearly, given a bi-partition (G_R, G_C) *consistent with direct determination*, we can trivially construct matrices $W \in \Re^{m\times|G_R|}, V \in \Re^{n\times|G_C|}$ such that A can be directly determined from $(W^T A, AV)$.

If we relax the restriction that each nonzero element of J be determined directly then it is possible that the work required to evaluate the nonzeroes of J can be further reduced. For example we could allow for a "substitution" process when recovering the nonzeroes of J from the pair $(W^T J, JV)$. Figures 3.1, 3.2 illustrate that a substitution method can win over direct determination: Figure 3.1 corresponds to direct determination, Figure 3.2 corresponds to determination using substitution.

$$\begin{pmatrix}
\square_{11} & \triangle_{12} & \triangle_{13} & & & & & & \\
\square_{21} & & & & & & & & \\
\square_{31} & & & & & & & & \\
\square_{41} & & & \square_{44} & \triangle_{45} & \triangle_{46} & & & \\
 & & & \square_{54} & & & & & \\
 & & & \square_{64} & & & & & \\
 & & & \square_{74} & & & \square_{77} & \triangle_{78} & \triangle_{79} \\
 & & & & & & \square_{87} & & \\
 & & & & & & \square_{97} & & \\
 & & & & & & \square_{10,7} & &
\end{pmatrix}$$

Figure 3.1: Optimal partition for direct method

In both cases elements labelled $\square$ are computed from the column grouping, i.e., calculated using the product JV; elements labelled $\triangle$ are calculated form the row groupings, i.e., calculated using the product $W^T J$. The matrix in Figure 3.1 indicates that we can choose G_C with $|G_C| = 2$ and G_R with $|G_R| = 1$ and determine all elements directly. That is, choose $V = (e_1 + e_7, e_4)$; choose $W = (e_1 + e_4 + e_7)$. Therefore in this case the work to compute J satisfies $\omega(J) \sim 3 \cdot \omega(F)$. Note that some elements can be determined twice, e.g., J_{11}.

However, the matrix in Figure 3.2 shows how to obtain the nonzeroes of J, using substitution, in work proportional to $2 \cdot \omega(F)$. Let $V = W = (e_1 + e_4 + e_7)$. Let p_1 be the (forward) computed

$$\begin{pmatrix}
\square_{11} & \triangle_{12} & \triangle_{13} & & & & & & \\
\square_{21} & & & & & & & & \\
\square_{31} & & & & & & & & \\
\square_{41} & & & \triangle_{44} & \triangle_{45} & \triangle_{46} & & & \\
 & & & \square_{54} & & & & & \\
 & & & \square_{64} & & & & & \\
 & & & \square_{74} & & & \triangle_{77} & \triangle_{78} & \triangle_{79} \\
 & & & & & & \square_{87} & & \\
 & & & & & & \square_{97} & & \\
 & & & & & & \square_{10,7} & &
\end{pmatrix}$$

Figure 3.2: Optimal partition for substitution method

vector $p_1 = JV$; let p_2^T be the (reverse) computed row vector $p_2^T = W^T J$. Then

$$p_1 = \begin{pmatrix} \square_{11} \\ \square_{21} \\ \square_{31} \\ \square_{41} + \triangle_{44} \\ \square_{54} \\ \square_{64} \\ \square_{74} + \triangle_{77} \\ \square_{87} \\ \square_{97} \\ \square_{10,7} \end{pmatrix}, \quad p_2^T = \left(\triangle_{11} + \square_{41}, \triangle_{12}, \triangle_{13}, \triangle_{44} + \square_{74}, \triangle_{45}, \triangle_{46}, \triangle_{77}, \triangle_{78}, \triangle_{79}\right).$$

Most of the nonzero elements are determined directly (no conflict). The remaining elements can be resolved,

$$\square_{74} = p_1(7) - p_2(7); \qquad \triangle_{44} = p_2(4) - \square_{74}; \qquad \square_{41} = p_1(4) - \triangle_{44}.$$

It is easy to extend this example so that the difference between the number of groups needed, between substitution and direct determination, increases with the dimension of the matrix. For example, a block generalization is illustrated in Figure 3.3: if we assume $l > 2w$ it is straightforward to verify that in the optimal partition the number of groups needed for direct determination will be $3w$ and determination by substitution requires $2w$ groups.

Definition 3.3 *A bi-partition* (G_R, G_C) *of a matrix* A *is* **consistent with determination by substitution**, *if there exists an ordering* π *on elements* a_{ij}*, such that for every nonzero* a_{ij} *of* A*, either column* j *is in a group where all nonzeros in row* i*, from other columns in the group, are ordered lower than* a_{ij}*, or row* i *is in a group where all the nonzeros in column* j*, from other rows in the group, are ordered lower than* a_{ij}*.*

In the usual way we can construct a matrix V from the column grouping G_C and a matrix W from the row grouping G_R: for example, to construct the columns of V associate with each group

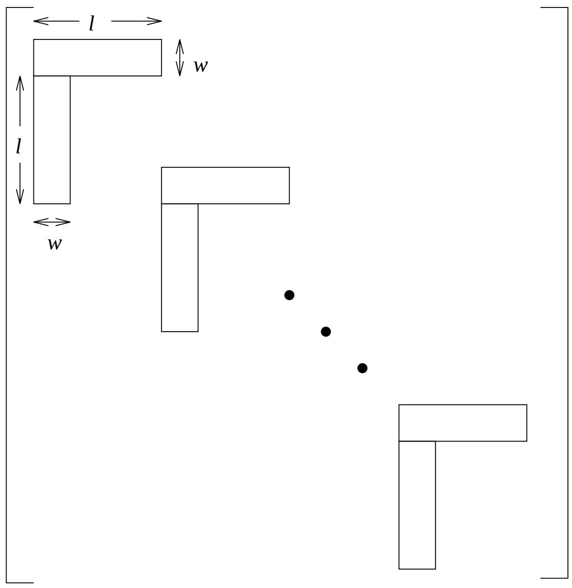

Figure 3.3: Block example

in G_C a boolean vector, with unit entries indicating membership of the corresponding columns. We can now state our main problem(s) more precisely:

> *The bi-partition problem (direct)* : Given a matrix A, obtain a bi-partition (G_R, G_C) consistent with direct determination, such that total number of groups, $|G_R| + |G_C|$, is minimized.
>
> *The bi-partition problem (substitution)* : Given a matrix A, obtain a bi-partition (G_R, G_C) consistent with determination by substitution, such that total number of groups, $|G_R| + |G_C|$, is minimized.

The bi-partition problems can also be expressed in terms of graphs and graph coloring. This graph view is important in that it more readily exposes the relationship of the bi-partition problems with the combinatorial approaches used in the sparse finite-differencing literature, e.g., [4, 5, 6, 7, 8]. However, we note that the remainder of this paper, with the exception of the error analysis in §6, does not rely directly on this graph interpretation.

To begin, we need the usual notion of a coloring of the vertices of a graph, the definition of a bipartite graph, and the concept of *path* coloring [4, 7] specialized to the bipartite graph case.

A p-coloring of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices or nodes and $\mathcal{E}$ is the set of edges, is a function

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

such that $\phi(u) \neq \phi(v)$, if $(u, v) \in \mathcal{E}$. The *chromatic number* $\chi(\mathcal{G})$ is the smallest p for which $\mathcal{G}$ has a p-coloring. A p-coloring ϕ of $\mathcal{G}$ induces a partition of vertices $\mathcal{V}$ into p groups $G_k, k = 1, 2, \ldots, p$, such that

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

Given a matrix $A \in \Re^{m\times n}$, define a bipartite graph $\mathcal{G}_b(A) = ([\mathcal{V}_1, \mathcal{V}_2], \mathcal{E})$ where $\mathcal{V}_1 = \{r_1, r_2, \ldots, r_n\}$, $\mathcal{V}_2 = \{c_1, c_2, \ldots, c_n\}$, c_j corresponds to the jth column of A, and r_i corresponds to the ith row of A. There is an edge, $(r_i, c_j) \in \mathcal{E}$ if a_{ij} is a nonzero in A.

In [4, 7] a *path p-coloring* of a graph is defined to be a vertex coloring using p colors with the additional property that every path of at least 3 *edges* uses at least 3 colors. Here we need a slight modification of that concept appropriate for the direct determination problem. We note that "color 0" is distinguished in that it corresponds to the lack of a true color assignment: i.e., $\phi(i) = 0$ indicates that vertex i is not assigned a color.

Definition 3.4 *Let $\mathcal{G}_b = ([\mathcal{V}_1, \mathcal{V}_2], \mathcal{E})$ be a bipartite graph. A mapping $\phi : [\mathcal{V}_1, \mathcal{V}_2] \to \{0, 1, \ldots, p\}$ is a* **bipartite path p-coloring** *of $\mathcal{G}_b$ if*

1. *Adjacent vertices have different assignments, i.e., if $(i, j) \in \mathcal{E}$ then $\phi(i) \neq \phi(j)$.*

2. *The set of* **positive** *colors used by vertices in $\mathcal{V}_1$ is disjoint from the set of* **positive** *colors used by vertices in $\mathcal{V}_2$: i.e., $i \in \mathcal{V}_1$, $j \in \mathcal{V}_2 \Rightarrow \{\phi(i) \neq \phi(j)$ **or** $\phi(i) = \phi(j) = 0\}$.*

3. *If vertices i and j are adjacent to vertex k with $\phi(k) = 0$, then $\phi(i) \neq \phi(j)$.*

4. *Every path of 3* **edges** *uses at least at least 3 colors.*

The smallest number for which graph $\mathcal{G}_b$ is bipartite path p-colorable is denoted by $\chi_p(\mathcal{G}_b)$. Figure 3.4 shows a valid bipartite path p-coloring. Numbers adjacent to the vertices denote colors. We note that Hossain and Steihaug [15] define a similar concept. However, their definition of path p-coloring does not allow for the "uncolor assignment", i.e., $\phi(i) = 0$. Consequently, a technique to remove empty groups is needed [15].

We are now in position to state the graph analogy to the concept of a bi-partition consistent with direct determination.

Theorem 3.1 *Let A be a $m \times n$ matrix with corresponding bipartite graph $\mathcal{G}_b(A) = ([\mathcal{V}_1, \mathcal{V}_2], \mathcal{E})$. The mapping $\phi : [\mathcal{V}_1, \mathcal{V}_2] \to \{0, 1, \ldots, p\}$ induces a bi-partition (G_R, G_C), with $|G_R| + |G_C| = p$, consistent with direct determination if and only if ϕ is a bipartite path p-coloring of $\mathcal{G}_b(A)$.*

Proof. ($\Leftarrow$) Assume that ϕ is a bipartite path p-coloring of $G_b(A)$, inducing a bi-partition (G_R, G_C) of rows and columns of A. If this bi-partition is not consistent with direct determination, then there is a nonzero element a_{ij} in the matrix for which the definition "either column j is in a group of G_C which has no other column having a nonzero in row i, or row i is in a group of G_R which has no other rows having a nonzero in column j" doesn't hold. This can happen only if one of the following cases hold:

- $\phi(r_i) = 0, \phi(c_j) \neq 0$ and there exists a column q with $a_{iq} \neq 0$, such that $\phi(c_j) = \phi(c_q)$. But this contradicts condition 3.

- $\phi(c_j) = 0, \phi(r_i) \neq 0$ and there exists a row p with $a_{pj} \neq 0$, such that $\phi(r_i) = \phi(r_p)$. But this contradicts condition 3.

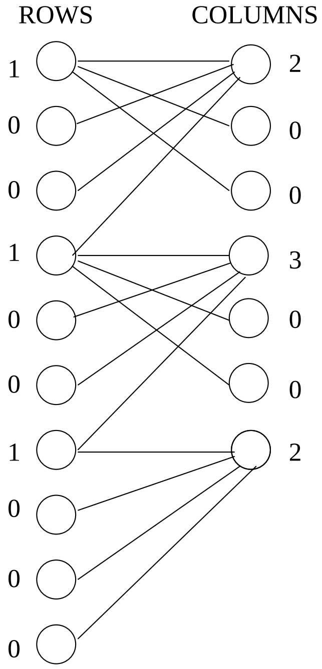

Figure 3.4: A valid bipartite path coloring

- $\phi(r_i) \neq 0, \phi(c_j) \neq 0$. 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 ϕ is a 2-coloring of the path $(r_p - c_j - r_i - c_q)$, a contradiction of condition 4.

($\Rightarrow$) Conversely, assume that ϕ induces a bi-partition consistent with direct determination of A. It is clear that conditions $1-3$ must be satisfied. It remains for us to establish condition 4: i.e., every path of 3 edges uses at least 3 colors. Suppose there is a bi-colored path: $r_i - c_j - r_k - c_l$, where $\phi(r_i) = \phi(r_k)$, $\phi(c_j) = \phi(c_l)$. Clearly by condition 3 the two colors on this path are positive. It is easy to see that element a_{jk} cannot be determined directly: there is a conflict in row group $\phi(r_i) = \phi(c_j)$ and there is a conflict in column group $\phi(c_k) = \phi(c_l)$, and these are the only two chances to determine a_{jk}. $\square$.

To capture the substitution notion the cyclic p-coloring definition [4] is modified slightly and applied to a bipartite graph.

Definition 3.5 *Let A be a $m \times n$ matrix with corresponding bipartite graph $\mathcal{G}_b(A) = ([\mathcal{V}_1, \mathcal{V}_2], \mathcal{E})$. A mapping $\phi : [\mathcal{V}_1, \mathcal{V}_2] \to \{0, 1, \ldots, p\}$ is a* **bipartite cyclic p-coloring** *of $\mathcal{G}_b$ if*

1. *Adjacent vertices have different assignments, i.e., if $(i,j) \in \mathcal{E}$ then $\phi(i) \neq \phi(j)$.*
2. *The set of* **positive** *colors used by vertices in $\mathcal{V}_1$ is disjoint from the set of* **positive** *colors used by vertices in $\mathcal{V}_2$: i.e., $i \in \mathcal{V}_1$, $j \in \mathcal{V}_2 \Rightarrow \{\phi(i) \neq \phi(j)$* **or** *$\phi(i) = \phi(j) = 0\}$.*
3. *If vertices i and j are adjacent to vertex k with $\phi(k) = 0$, then $\phi(i) \neq \phi(j)$.*
4. *Every cycle uses at least at least 3 colors.*

The smallest number for which graph $\mathcal{G}_b$ is bipartite cyclic p-colorable is denoted by $\chi_c(\mathcal{G}_b)$. Figure 3.5 shows a valid bipartite cyclic p-coloring note that only 2 colors are necessary whereas the bipartite path p-coloring in Figure 3.4 requires 3 colors. The notion of a bi-partition consistent

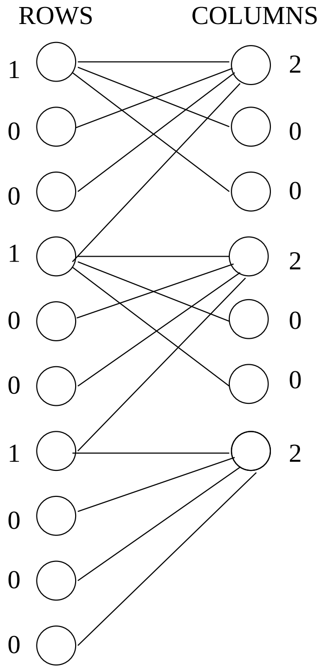

Figure 3.5: A valid bipartite cyclic coloring

with determination via substitution can now be cleanly started in graph-theoretic terms.

Theorem 3.2 *Let A be a $m \times n$ matrix with corresponding bipartite graph $\mathcal{G}_b(A) = ([\mathcal{V}_1, \mathcal{V}_2], \mathcal{E})$. The mapping $\phi : (\mathcal{V}_1, \mathcal{V}_2) \to \{0, 1, \ldots, p\}$ induces a bi-partition (G_R, G_C), with $|G_R| + |G_C| = p$, consistent with determination by substitution if and only if ϕ is a bipartite cyclic p-coloring of $\mathcal{G}_b(A)$.*

Proof. ($\Rightarrow$) Assume ϕ induces a bi-partition consistent with determination by substitution but ϕ is not a bipartite cyclic p-coloring of $G_b(A)$. Clearly condition $1 - 2$ must hold; it is easy to see that if condition 3 doesn't hold then not all nonzero elements can be determined. The only non-trivial violation is condition 4: there is a cycle which has only two colors, i.e all the vertices $\in V_1$ in the cycle have the same color c_1, and all the vertices $\in V_2$ in the cycle have the same color c_2. Note that neither c_1 and c_2 can be equal to 0, since a node colored 0 in a cycle would imply that its adjacent vertices are both colored differently, implying that there are at least 3 colors. Consider the submatrix A_s of A, corresponding to this cycle. Submatrix A_s has at least two non zeros in each row and in each column, since each vertex has degree 2 in the cycle. But since we are considering substitution methods only, at least one element of A_s needs to be computed directly. Clearly there is no way to get any element of this submatrix directly, a contradiction.

($\Leftarrow$) Conversely, assume that ϕ is a bipartite cyclic p-coloring of $G_b(A)$ but that bi-partition (G_R, G_C) induced by ϕ is not consistent with determination by substitution. But, edges (nonzeros) with one end assigned color “0” can be determined directly: by the definition of bi-coloring there will be no conflict. Moreover, every pair of positive colors induces a forest (i.e., a collection of trees); therefore, the edges (nonzeros) in the induced forest can be resolved via substitution [4]. □

The two bi-partition problems can now be simply stated in terms of optimal bipartite path and cyclic p-colorings:

The bipartite path p-coloring problem : Determine a bipartite path p-coloring of $\mathcal{G}_b(A)$ with the smallest possible value of p, i.e., $p = \chi_p(\mathcal{G}_b)$.
The bipartite cyclic p-coloring problem : Determine a bipartite cyclic p-coloring of $\mathcal{G}_b(A)$ with the smallest possible value of p, i.e., $p = \chi_c(\mathcal{G}_b)$.

The graph theoretic view is useful for both analyzing the complexity of the combinatorial problem and suggesting possible algorithms, exact or heuristic. In fact, using the the p-coloring notions discussed above, and an approach similar to that taken in [4], it is easy to show that corresponding decision problems are NP-complete.

Bipartite cyclic p-coloring decision problem (CCDP): Given an integer $p \geq 3$ and an arbitrary bipartite graph $\mathcal{G}$, is it possible to assign a cyclic p-coloring to nodes of $\mathcal{G}$?

Bipartite path p-coloring decision problem (PCDP) : Given an integer $p \geq 3$ and an arbitrary bipartite graph $\mathcal{G}$, is it possible to assign a bipartite path p-coloring to nodes of $\mathcal{G}$?

The proofs are a straightforward adaptation of those in [4] and we omit them here. The upshot of these (negative) complexity results is that in practise we must turn our attention to (fast) heuristics to approximately solve the cyclic and path coloring problems. In the next section we present simple, effective, and “easy-to-visualize” heuristics for these two combinatorial problems.

Finally, it is easy to establish a partial ordering of chromatic numbers:

$$\chi_c(\mathcal{G}_b(A)) \leq \chi_p(\mathcal{G}_b(A)) \leq min(\chi(\mathcal{G}(A^T)), \chi(\mathcal{G}(A))), \tag{3.2}$$

where $\mathcal{G}(M)$ refers to the column intersection graph of matrix M, $\chi(\mathcal{G}(M))$ is the (usual) chromatic number of graph $\mathcal{G}(M)$.

The first inequality in (3.2) holds because if ϕ is a bipartite path p-coloring then ϕ is a bipartite cyclic p-coloring; the second inequality holds because a trivial way to satisfy conditions $1-4$ of Definition 3.4 is to assign “0” to all the row (column) nodes and then use positive colors on all the column (row) nodes to satisfy condition 3. This ordering supports the tenet that use of bi-partition/bi-coloring is never worse than one-sided calculation and that a substitution approach is never worse than a direct approach (in principle).

4 Bi-coloring

The two combinatorial problems we face, corresponding to direct determination and determination by substitution, can both be approached in the following way. First, permute and partition the structure of J: $\tilde{J} = P \cdot J \cdot Q = [J_C|J_R]$, as indicated in Figure 4.1. The construction of this partition is crucial; however, we postpone that discussion until after we illustrate its' utility. Assume $P = Q = I$ and $J = [J_C|J_R]$.

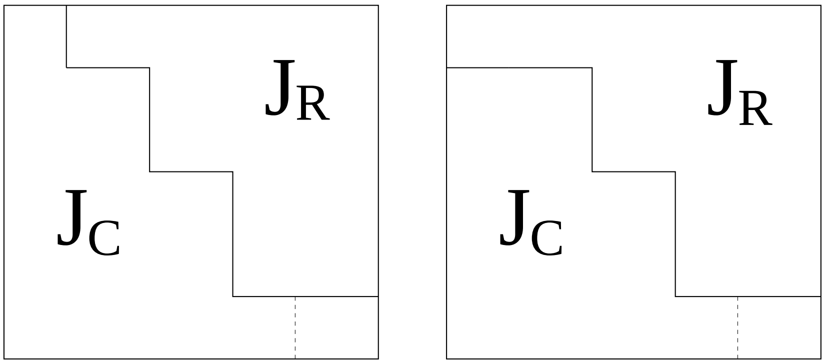

Figure 4.1: Possible partitions of the matrix $\tilde{J} = P \cdot J \cdot Q$

Second, define appropriate intersection graphs $\mathcal{G}_C^I, \mathcal{G}_R^I$ based on the partition $[J_C|J_R]$; a coloring of $\mathcal{G}_C^I$ yields a partition of a subset of the columns, G_C, which defines matrix V. Matrix W is defined by a partition of a subset of rows, G_R, which is given by a coloring of $\mathcal{G}_R^I$. We call this double coloring approach *bi-coloring*. The difference between the direct and substitution cases is in how the intersection graphs, $\mathcal{G}_C^I, \mathcal{G}_R^I$, are defined, and how the nonzeroes of J are extracted from the pair $(W^T J, JV)$.

4.1 Direct determination

In the direct case the intersection graph $\mathcal{G}_C^I$ is defined: $\mathcal{G}_C^I = (\mathcal{V}_C^I, \mathcal{E}_C^I)$ where

- Vertex $j \in \mathcal{V}_C^I$ if nnz[2]$(\text{column } j \cap J_C) \neq 0$.
- $(r, s) \in \mathcal{E}_C^I$ if $r \in \mathcal{V}_C^I, s \in \mathcal{V}_C^I$, $\exists k$ such that $J_{kr} \neq 0, J_{ks} \neq 0$ and **either** $(k, r) \in J_C$ **or** $(k, s) \in J_C$.

The key point in the construction of graph $\mathcal{G}_C^I$, and why $\mathcal{G}_C^I$ is distinguished from the usual column intersection graph, is that columns r and s are said to intersect if and only if their nonzero locations overlap, in part, in J_C: i.e., columns r and s intersect if $J_{kr} \cdot J_{ks} \quad = 0$ and **either** $(k, r) \in J_C$ **or** $(k, s) \in J_C$ for some k.

The "transpose" of the procedure above is used to define $\mathcal{G}_R^I = (\mathcal{V}_R^I, \mathcal{E}_R^I)$. Specifically, $\mathcal{G}_R^I = (\mathcal{V}_R^I, \mathcal{E}_R^I)$ where

- Vertex $i \in \mathcal{V}_R^I$ if $nnz(\text{row } i \cap J_R) \neq 0$.
- $(r, s) \in \mathcal{E}_R^I$ if $r \in \mathcal{V}_R^I, s \in \mathcal{V}_R^I$, $\exists k$ such that $J_{rk} \neq 0, J_{sk} \neq 0$ and **either** $(r, k) \in J_R$ **or** $(s, k) \in J_R$.

[2]If M is a matrix or a vector then "nnz(M)" is the number of nonzeroes in M

In this case the reason graph $\mathcal{G}_R^I$ is distinguished from the usual row intersection graph is that rows r and s are said to intersect if and only if their nonzero locations overlap, in part, in J_R: rows r and s intersect if $J_{rk} \cdot J_{sk} \ = 0$ and **either** $(r,k) \in J_R$ **or** $(s,k) \in J_R$ for some k.

The bi-partition (G_R, G_C), induced by coloring of graphs $\mathcal{G}_R^I$ and $\mathcal{G}_C^I$, is consistent with direct determination of J. To see this consider a nonzero element $(i,j) \in J_C$: column j is in a group of G_C (corresponding to a color) with the property that no other column in G_C has a nonzero in row i: hence, element (i,j) can be directly determined. Analagously, consider a nonzero element $(r,s) \in J_R$: row r will be in a group of G_R (corresponding to a color) with the property that no other row in G_R has a nonzero in column s: hence, element (r,s) can be directly determined. Since every nonzero of J is covered, the result follows.

Example: Consider the example Jacobian matrix structure shown in Figure 4.2 with the partition (J_C, J_R) shown.

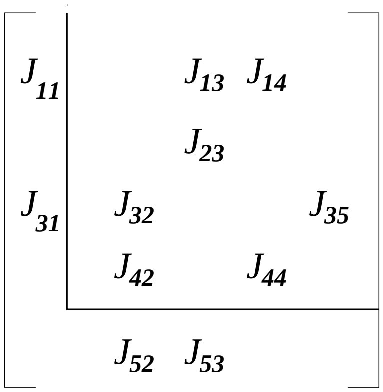

Figure 4.2: Example Partition

The graphs $\mathcal{G}_C$ and $\mathcal{G}_R$ formed by the algorithm outlined above are given in Figure 4.3. Coloring $\mathcal{G}_C$ requires 3 colors, while $\mathcal{G}_R$ can be colored in two. Boolean matrices V and W can be formed in the usual way: each column corresponds to a group (or color) and unit entries indicate column (or row) membership in that group:

$$V = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad JV = \begin{pmatrix} J_{11} & 0 & \times \\ 0 & 0 & \times \\ J_{31} & \times & \times \\ 0 & \times & 0 \\ 0 & J_{52} & J_{53} \end{pmatrix}$$

$$W = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \qquad W^T J = \begin{pmatrix} \times & J_{32} & J_{13} & J_{14} & J_{35} \\ \times & J_{42} & J_{23} & J_{44} & 0 \end{pmatrix}.$$

Clearly, all nonzero entries of J can be identified in either JV or $W^T J$.

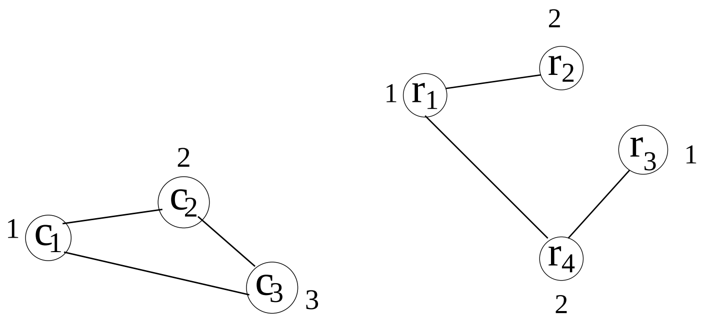

Figure 4.3: Graphs $\mathcal{G}_C$ and $\mathcal{G}_R$ (direct approach)

4.2 Determination by substitution

The basic advantage of determination by substitution in conjunction with partition $J = [J_C|J_R]$ is that sparser intersection graphs $\mathcal{G}_C^I, \mathcal{G}_R^I$ can be used. Sparser intersection graphs mean thinner matrices V,W which, in turn, result in reduced cost.

In the substitution case the intersection graph $\mathcal{G}_C^I$ is defined: $\mathcal{G}_C^I = (\mathcal{V}_C^I, \mathcal{E}_C^I)$ where

- Vertex $j \in \mathcal{V}_C^I$ if nnz (column $j \cap J_C) \neq 0$.
- $(r, s) \in \mathcal{E}_C^I$ if $r \in \mathcal{V}_C^I, s \in \mathcal{V}_C^I$, $\exists k$ such that $J_{kr} \neq 0, J_{ks} \neq 0$ and **both** $(k, r) \in J_C$, $(k, s) \in J_C$.

Note that the intersection graph $\mathcal{G}_C^I = (\mathcal{V}_C^I, \mathcal{E}_C^I)$ captures the notion of two columns intersecting if there is overlap in nonzero structure in J_C: columns r and s intersect if $J_{kr} \cdot J_{ks} \ = 0$ and **both** $(k, r) \in J_C$, $(k, s) \in J_C$ for some k. It is easy to see that $\mathcal{E}_C^I$ is a subset of the set of edges used in the direct determination case.

The "transpose" of the procedure above is used to define $\mathcal{G}_R^I = (\mathcal{V}_R^I, \mathcal{E}_R^I)$. Specifically, $\mathcal{G}_R^I = (\mathcal{V}_R^I, \mathcal{E}_R^I)$ where

- Vertex $i \in \mathcal{V}_R^I$ if row $i \in J_R$ and nnz(row $i \cap J_R) \neq 0$.
- $(r, s) \in \mathcal{E}_R^I$ if $r \in \mathcal{V}_R^I, s \in \mathcal{V}_R^I$, $\exists k$ such that $J_{rk} \neq 0, J_{sk} \neq 0$ and **both** $(r, k) \in J_R$, $(s, k) \in J_R$.

The intersection graph $\mathcal{G}_R^I = (\mathcal{V}_R^I, \mathcal{E}_R^I)$ captures the notion of two rows intersecting if there is overlap in nonzero structure in J_R: rows r and s intersect if $J_{rk} \cdot J_{sk} \ = 0$ and $(r, k) \in J_R$ **and** $(s, k) \in J_R$ for some k. It is easy to see that $\mathcal{E}_R^I$ is a subset of the set of edges used in direct determination.

All the elements of J can be determined from $(W^T J, JV)$ by a substitution process. This is evident from the illustrations in Figure 4.4.

Figure 4.4 illustrates two of four possible nontrival types of partitions. In both cases it is clear that nonzero elements in the section labelled "1" can be solved for directly – by the construction process they will be in different groups. Nonzero elements in "2" can either be determined directly, or will depend on elements in section "1". But elements in section "1" are already determined (directly) and so, by substitution, elements in "2" can be determined after "1". Elements in section "3" can then be determined, depending only on elements in "1" and "2", and so on until the entire matrix is resolved.

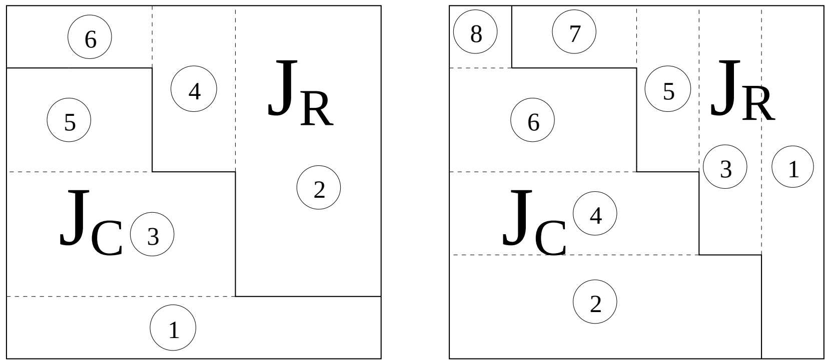

Figure 4.4: Substitution Orderings

Example. Consider again the example Jacobian matrix structure shown in Figure 4.2. Column and row intersection graphs corresponding to substitution are given in Figure 4.5. Note that $\mathcal{G}_C$ is disconnected and requires 2 colors; $\mathcal{G}_R$ is a simple chain and also requires 2 colors.

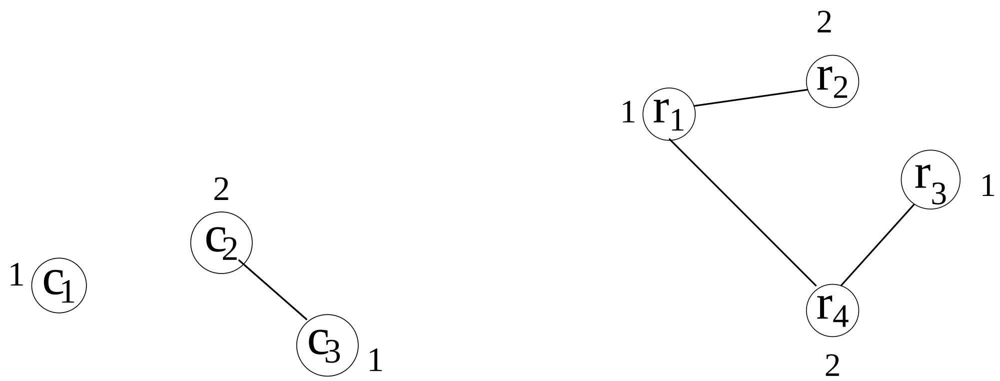

Figure 4.5: Graphs $\mathcal{G}_C$ and $\mathcal{G}_R$ for substitution process

The coloring of $\mathcal{G}_C$ and $\mathcal{G}_R$ leads to the following matrices V, W and the resulting computation of JV, W^TJ:

$$V = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \qquad JV = \begin{pmatrix} J_{11} + J_{13} & 0 \\ \times & 0 \\ J_{31} & \times \\ 0 & \times \\ J_{53} & J_{52} \end{pmatrix}$$

$$W = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \qquad W^TJ = \begin{pmatrix} \times & J_{32} & J_{13} & J_{14} & J_{35} \\ \times & J_{42} & J_{23} & J_{44} & 0 \end{pmatrix}$$

It is now easy to verify that all nonzeroes of J can be determined via substitution.

4.3 How to partition J.

We now consider the problem of obtaining a useful partition $[J_C|J_R]$, and corresponding permutation matrices P,Q, as illustrated in Figure 4.1. A simple heuristic is proposed based on the knowledge that the subsequent step, in both the direct and the substitution method, is to color intersection graphs based on this partition.

Algorithm **MNCO** builds partition J_C from bottom up, and partition J_R from right to left. At the k^{th} major iteration either a new row is added to J_C or a new column is added to J_R: the choice depends on considering a lower bound effect:

$$\rho(J_R^T) + max(\rho(J_C), nnz(r)) < (\rho(J_C) + max(\rho(J_r^T), nnz(c)),$$

where $\rho(A)$ is the maximum number of nonzeroes in any row of matrix A, r is a row under consideration to be added to J_C, c is a column under consideration to be added to J_R. Hence, the number of colors needed to color $\mathcal{G}_C^I$ is bounded below by $\rho(J_C)$; the number of colors needed to color $\mathcal{G}_R^I$ is bounded below by $\rho(J_R^T)$.

In algorithm **MNCO**, matrix $M = J(R,C)$ is the submatrix of J defined by row indices R and column indices C: M consists of rows and columns of J not yet assigned to either J_C or J_R.

Minimum Nonzero Count Ordering (MNCO)

1. Initialize $R = (1:m)$, $C = (1:n)$, $M = J(R,C)$
2. Find $r \in R$ with fewest nonzeros in M
3. Find $c \in C$ with fewest nonzeros in M
4. Repeat Until $M = \emptyset$

    if $\rho(J_R^T) + max(\rho(J_C), nnz(r)) < (\rho(J_C) + max(\rho(J_R^T), nnz(c))$ (LB)

        J_C=J_C ∪ (r ∩ C)

        R=R-{r}

    else

        J_R=J_R ∪ (c ∩ R)

        C=C-{c}

    end if

    $M = J(R,C)$.

   end repeat

Note that, upon completion, J_R, J_C have been defined; the requisite permutation matrices are implicitly defined by the ordering chosen in **MNCO**.

5 Bi-coloring performance

In this section we present results of numerical experiments. The work required to compute the sparse Jacobian matrix is the work needed to compute $(W^T J, JV)$ which, in turn, is proportional to the work to evaluate the function F times the sum of the column dimensions of the boolean matrices $V \in \Re^{n \times t_V}$, $W \in \Re^{m \times t_W}$. The column dimension sum, $t_V + t_W$, is equal to the number of colors used in the bi-coloring. In our experiments we compare the computed coloring numbers required for the direct and substitution approaches. We also compute the number of colors required by one-sided schemes: a column partition alone corresponds to the construction of V based on coloring the column intersection graph of J, a row partition alone corresponds to the construction of W based on coloring the row intersection graph of J. The latter case leads to the application of the reverse mode of AD (alone), whereas the former case leads to use of the forward mode.

Both the direct and substitution methods require colorings of their respective pairs of intersection graphs, $\mathcal{G}_C^I$, $\mathcal{G}_R^I$. Many efficient graph coloring heuristics are available: in our experiments we use the incidence degree (ID) ordering [3, 8].

We use three sources of test matrices: a linear programming testbed with results reported in Table 1 and summarized in Table 2; the Harwell-Boeing sparse matrix collection, with results reported in Tables 3,4; self-generated m-by-n "grid matrices" with results given in Tables 5, 6. A grid matrix is constructed in the following way. First, approximately $\sqrt{n}$ of the columns are chosen, spaced uniformly. Each chosen column is randomly assigned $DENS \cdot m$ nonzeroes. Second, approximately $\sqrt{m}$ of the rows are chosen, spaced uniformly. Each chosen row is randomly assigned $DENS \cdot n$ nonzeroes. We vary $DENS$ as recorded in Table 5.

For each problem we cite the dimensions of the matrix A and the number of nonzeros(nnz). The experimental results we report are the number of colors required by our bi-coloring approach, both direct and substitution, and the number of colors required by one-sided schemes.

				Bi-coloring		One-sided	
Name	m	n	nnz	Direct	Substitution	column	row
standata	359	1274	3230	9	7	745	10
scagr25	471	671	1725	8	5	10	9
scagr7	129	185	465	8	5	10	9
stair	356	620	4021	36	29	36	36
blend	74	114	522	16	14	29	16
vtp.base	198	347	1052	12	10	38	12
agg	488	615	2862	19	13	43	19
agg2	516	758	4740	26	21	49	43
agg3	516	758	4756	27	21	52	43
bore3d	233	334	1448	28	24	73	28
israel	174	316	2443	61	49	119	136
boeing1	351	726	3827	32	28	315	32
boeing2	166	305	1358	23	18	93	23
tuff	333	630	4563	21	16	114	25
adlittle	56	138	424	11	10	27	11

Table 1: LP Constraint Matrices (http://www.netlib.org/lp/data/)

Bi-coloring		1-sided Coloring	
Direct	Substitution	column	row
337	270	1753	452

Table 2: Totals for LP Collection

				Bi-coloring		1-sided Coloring	
Name	M	N	NNZ	Direct	Substitution	column	row
watt2	1856	1856	11550	20	12	128	65
cannes 256	256	256	2916	32	23	83	83
cannes 268	268	268	1675	18	12	33	33
cannes 292	292	292	2540	17	17	35	35
cannes 634	634	634	7228	28	21	28	28
cannes 715	715	715	6665	22	18	105	105
cannes 1054	1054	1054	12196	31	23	35	35
cannes 1072	1072	1072	12444	32	24	35	35
chemimp/impcolc	137	137	411	6	4	8	9
chemimp/impcold	425	425	1339	6	5	11	11
chemimp/impcole	225	225	1308	21	14	21	31
chemwest/west0067	67	67	294	9	7	9	12
chemwest/west0381	381	381	2157	12	9	29	50
chemwest/west0497	497	497	1727	22	19	28	55
smtape/gent113	113	113	655	19	13	20	27
smtape/arc130	130	130	1282	25	23	124	124

Table 3: The Harwell-Boeing collection (ftp from orion.cerfacs.fr)

5.1 Observations

First, we observe that the bi-coloring approach is often a significant win over one-sided determination. Occasionally, the improvement is spectacular, e.g., "cannes 715". Improvement in the Harwell-Boeing problems are generally more significant than on the LP collection in the sense that bi-coloring significantly outperforms both one-sided possibilities. This is partially due to the fact that the matrices in the LP collection are rectangular whereas the matrices in the Harwell-Boeing collection are square: calculation of the nonzeroes of J from $W^T J$ alone can be quite attractive when J has relatively few rows. The grid collection displays the advantage of bi-coloring to great effect - grid matrices are ideal bi-coloring candidates.

In general the advantage of substitution over direct determination is not as great as the difference between bi-coloring and one-sided determination. Nevertheless, fewer colors are almost always needed and for expensive functions F this can be important. For most problems the gain is about 20% though it can approach 50%, e.g., "watt2".

Bi-coloring		1-sided Coloring	
Direct	Substitution	column	row
320	244	732	738

Table 4: Totals for Harwell-Boeing Collection

			Bi-coloring		1-sided coloring	
M	N	DENS	Direct	Substitution	column	row
10	10	0.44	5	4	6	7
10	10	0.75	6	5	7	8
10	10	0.98	6	6	9	9
10	20	0.55	8	8	12	8
10	20	0.89	8	8	19	10
10	20	1.0	8	8	20	10
25	25	0.52	11	10	18	15
25	25	0.61	11	10	21	19
25	25	0.99	10	10	25	25
25	100	0.56	15	16	61	24
25	100	0.67	15	15	81	24
25	100	1.00	15	15	100	25
100	100	0.52	20	20	84	74
100	100	0.64	20	20	95	93
100	100	1.00	20	20	100	100
100	400	0.53	30	30	282	98
100	400	0.64	30	30	352	100
100	400	1.00	30	30	400	100

Table 5: Grid Matrices

5.2 Interface with ADOL-C

We have interfaced our coloring and substitution routines with the ADOL-C software. The C++ package ADOL-C [14] facilitates the evaluation of first and higher order derivatives of vector function, defined by programs written in C or C++.

We compare the time needed on a sample problem with respect to five approaches:

- AD/bi-coloring (direct)
- AD/bi-coloring (substitution)
- AD/column coloring (forward mode)
- AD/row coloring (reverse mode)
- FD (Sparse finite differencing based on column coloring)

The test function F we use is a simple nonlinear function: define $f(a) = a^4 + 5 \cdot a$, let $N(i)$ be the index set of nonzeroes in row i of the Jacobian matrix and define $F_i(x) = \sum_{j \in N(i)} f(x_j)$.

Bi-coloring		1-sided Coloring	
Direct	Substitution	column	row
268	265	1692	749

Table 6: Totals for Grid Matrices

The Jacobian matrix (and thus the sparsity pattern) is a 10-by-3 block version of Figure 3.3, i.e., $l = 10$, $w = 3$. Problem dimensions $n = 100, 200, 400$, as indicated in Figure 5.1, were used in the experiments.

Our results, portrayed in Figure 5.1, suggest the following order of execution time requirement by different techniques:

$$FD > AD/row > AD/column > AD/bi-coloring(direct) > AD/bi-coloring(substitution).$$

Note that FD requires more time than $AD/column$ even though the same coloring is used for both. This is because the work estimate $t_V \cdot \omega(F)$ is actually an upper bound on the work required by the forward mode where t_V is the number of columns of V. This bound is often loose in practise whereas $t_V \cdot \omega(F)$ is tight for finite-differencing since the subroutine to evaluate F is actually called (independently) t_V times.

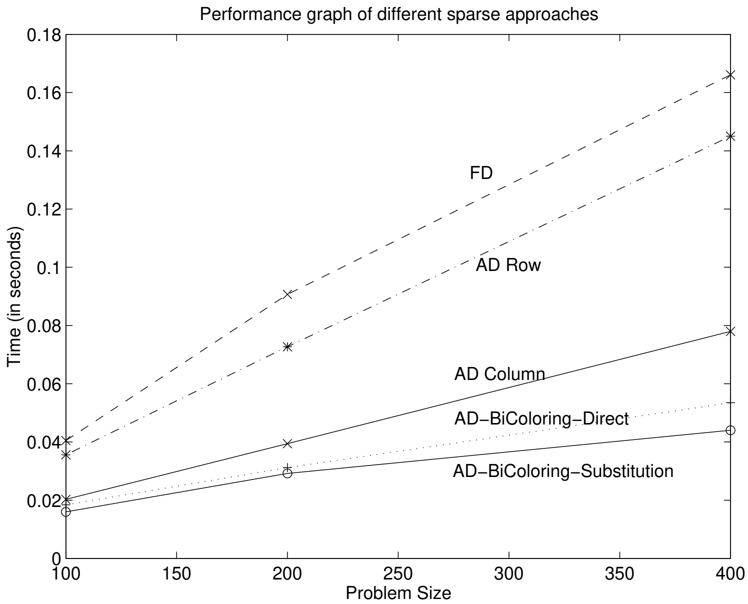

Figure 5.1: A comparison of different sparse techniques

Another interesting observation is that the reverse mode calculation ($AD/$Row) is about twice as expensive as the forward calculation ($AD/$Column). This is noteworthy because in this example, based on the structure Figure 3.1, the column dimensions of V and W are equal. This suggests that it may be practical to weigh the cost of the forward calclulation of Jv versus the calculation of $w^T J$, where w, v are vectors. We comment further on this aspect in §7.

6 Substitution and round-off

In general, the substitution approach requires fewer colors and therefore is more efficient[3], in principle, than direct determination. However, there is a possibility of increased round-off error due to the substitution process. In fact an analogous issue arises in the sparse Hessian approximation context [4, 7, 16] where, indeeed, there is considerable cause for concern. The purpose of this section is to examine this question in the AD context. The bottom line here is that there is less to worry about in this case. In the sparse Hessian approximation case significant error growth occurs when the finite-difference step size varies over the set of finite-difference directions; however, in our current setting there this is not an issue since the "step size" is equal to unity in all cases.

First we consider the number of substitutions required to determine any nonzero of J from $(W^T J, JV)$ where matrices W, V are chosen using our substitution stategy. There is good news: similar to the sparse Hessian approximation situation [4, 7, 16], the number of dependencies, or substitutions, to resolve a nonzero of J can be bounded above by $\frac{1}{2}\lfloor(m+n-2)\rfloor$.

Theorem 6.1 *Let ϕ be a bipartite cyclic p-coloring of $G_b(J)$. Then, ϕ corresponds to a substitution determination of J and each unknown in J is dependent on at most $m+n-2$ unknowns. Moreover, it is possible to order the calculations so that the maximum number of substitutions is less than or equal to $\frac{1}{2}\lfloor(m+n-2)\rfloor$.*

Proof : First, edges (nonzeros) with one end assigned color "0" can be determined directly: by the definition of bi-coloring there will be no conflict. Second, every pair of positive colors induces a forest (i.e., a collection of trees in $G_b(J)$); therefore, the edges (nonzeros) in the induced forest can be resolved via substitution [4]. Hence, all edges (nonzeros) can be resolved either directly or by a substitution process and the worst case corresponds to a tree with $m+n-1$ edges yielding an upper bound of $m+n-2$ substitutions. However, it is easy to see that the substitution calculations can be ordered to halve the worst-case bound yielding at most $\frac{1}{2}\lfloor(m+n-2)\rfloor$ substitutions. $\square$

Next we develop an expression to bound the error in the computed Jacobian. Except for the elements that can be resolved directly, the nonzero elements of the Jacobian matrix can be solved for by considering each subgraph induced by 2 positive colors (directions), one color corresponding to a subset of rows, one color corresponding to a subset of columns. Let us look at the subgraph $\mathcal{G}_{p,q}$ induced by colors p (columns), q (rows). Let $z_p = Jv_p$, $y_q^T = w_q^T J$, R_q be the set of rows colored q, and C_p be the set of columns colored p, where

$$w_q = \sum_{i \in R_q} e_i, \; v_p = \sum_{i \in C_p} e_i.$$

Let

$$\hat{z_p} = z_p + \epsilon_p^c, \qquad \hat{y_q} = y_q + \epsilon_q^r$$

denote the quantities computed via AD. Note that the errors introduced here are only due to the automatic differentiation process and are typically very small.

In the solution process an element J_{ij} is determined:

$$J_{ij} = \hat{z}_{p_i} - \sum_{k \in N(r_i)} J_{ik}, \; or \; J_{ij} = \hat{y}_{q_j} - \sum_{k \in N(c_j)} J_{kj}$$

[3] Of course a substitution method does incur the extra cost of performing the substitution calculation. However, this can be done very efficiently and the subsequent cost is usually negligible.

depending on whether a column equation (of form Jv) or a row equation (of form $w^T J$) is used. Here, $N(r_i)$ denotes set of neighbours of row i in $\mathcal{G}_{p,q}$, and $N(c_j)$ denotes set of neighbours of column j in $\mathcal{G}_{p,q}$.

Assume that $J^{actual} = F'(x)$ denotes the actual Jacobian matrix; hence,

$$J_{ij}^{actual} = z_{p_i} - \sum_{k \in N(r_i)} J_{ik}^{actual}, \ or \ J_{ij}^{actual} = y_{q_j} - \sum_{k \in N(c_j)} J_{kj}^{actual}.$$

Define an error matrix $E = J - J^{actual}$ and let ϵ_{ij} be the difference

$$\epsilon_{ij} = \hat{z_{p_i}} - \sum_{k \in N(r_i) \cup \{j\}} J_{ik}, \ or \ \epsilon_{ij} = \hat{y}_{q_j} - \sum_{k \in N(c_j) \cup \{i\}} J_{kj}$$

depending on the way element J_{ij} was computed.

We take into account the effect of *rounding errors* by letting $\hat{\epsilon_{ij}}$ to be equal to ϵ_{ij} plus the contribution from rounding errors due to use of the equation that determines J_{ij}. We can now write

$$\hat{\epsilon_{ij}} = E_{ij} + \sum_{k \in N(r_i)} E_{ik}, \ or \ \hat{\epsilon_{ij}} = E_{ij} + \sum_{k \in N(c_j)} E_{kj} \tag{6.1}$$

again, depending on the way J_{ij} is calculated.

Moreover, we let ϵ_{max} be the constant :

$$\epsilon_{max} = max_{i,j} |\hat{\epsilon_{ij}}|$$

Note that ϵ_{max} has no contribution from step sizes, unlike results for finite- differencing [4, 16].

Theorem 6.2 *If J is obtained by our substitution process then*

$$|E_{ij}| \leq \frac{1}{2} \lfloor (m+n) \rfloor \cdot \epsilon_{max}$$

Proof: From equation (6.1),

$$\hat{\epsilon_{ij}} = E_{ij} + \sum_{k \in N(r_i)} E_{ik}, \tag{6.2}$$

or

$$\hat{\epsilon_{ij}} = E_{ij} + \sum_{k \in N(c_j)} E_{kj}. \tag{6.3}$$

Let us assume, without loss of generality, that equation (6.2) holds. This implies a bound

$$|E_{ij}| \leq |\hat{\epsilon}_{ij}| + \sum_{k \in N(r_i)} |E_{ik}|$$

But the same decomposition can be applied recursively to each E_{ik}, and using Theorem 6.1, the result follows. □

There are two positive aspects to Theorem 6.2. First, unlike the sparse finite-difference substitution method for Hessian matrices [4, 7, 16], there is no dependence on a variable "step size": in *AD* the "step size" is effectively uniformly equal to unity. Second, there is no cumulative dependence on $nnz(J)$ but rather just on the matrix dimensions, $m+n$. However, there is one unsatisfactory aspect of the bound in Theorem 6.2: the bound is expressed in terms of ϵ_{max}, but ϵ_{max} is not known to be restricted in magnitude. A similar situation arises in the [4, 7, 16]. Nevertheless, as illustrated in the example discussed below, ϵ_{max} is usually modest in practise.

We conclude this section with a small experiment where we inspect final accuracies of the computed Jacobian matrices. The test function F is a simple nonlinear function as described in §5.2.

In Table 7 "FD1" is the sparse finite difference computation [8] using a fixed stepsize $\alpha = \sqrt{\epsilon} \approx 10^{-8}$; "FD2" refers to the sparse finite difference computation [8] using a variable stepsize: α is uniformly varied in the range $[\frac{\sqrt{\epsilon}}{4}, 4 \cdot \sqrt{\epsilon}]$. The column labelled "Rel error" records $\|ERR\|_2$, where the nonzeros of ERR correspond to the nonzeros of J: for $J_{ij} \neq 0$, $ERR_{ij} = \frac{J_{ij}^{actual} - J_{ij}^{computed}}{J_{ij}^{actual}}$.

The general trends we observe are the following. First, similar to the results reported in [1] for forward-mode direct determination, the Jacobian matrices determined by our bi-coloring/AD approach are significantly and uniformly more accurate than the finite-difference approximations. This is true for both direct determination and the substitution approach. Second, the direct approach is uniformly more accurate than the substitution method; however, the Jacobian matrices determined via substitution are sufficiently accurate for most purposes, achieving at least 10 digits of accuracy and usually more. Finally, on these problem there is relatively little difference in accuracy between the fixed step ($FD1$) method and the variable step ($FD2$) method. However, as illustrated in [6], it is easy to construct examples where the variable step ($FD2$) approach produces unacceptable accuracy.

	Direct	Substitution	FD1	FD2
size	Rel error	Rel error	Rel error	Rel error
100	1.30×10^{-15}	7.63×10^{-13}	7.06×10^{-6}	2.06×10^{-5}
200	2.93×10^{-15}	4.19×10^{-12}	1.06×10^{-5}	6.72×10^{-5}
400	9.62×10^{-15}	2.02×10^{-11}	4.77×10^{-5}	9.81×10^{-5}
800	4.57×10^{-14}	1.45×10^{-10}	9.21×10^{-5}	8.49×10^{-4}

Table 7: Errors (sample nonlinear problem)

7 Concluding remarks

We have proposed an effective way to compute a sparse Jacobian matrix, J, using automatic differentiation. Our proposal uses a new graph technique, bi-coloring, to divide the differentiation work between the two modes of automatic differentiation, forward and reverse. The forward mode computes the product JV for a given matrix V; the reverse mode computes the product $W^T J$ for a given matrix W. We have suggested ways to choose thin matrices V, W so that the work to compute the pair $(W^T J, JV)$ is modest and so that the nonzero elements of J can be readily extracted.

Our numerical results strongly support the view that bi-coloring/AD is superior to one-sided computations (both AD and FD) with respect to the order of work required. Of course AD approaches offer additional advantages over FD schemes: significantly better accuracy, no need to heuristically determine a step size rule, and the sparsity pattern need not be determined a priori [14].

Implicit in our approach is the assumption that the cost to compute Jv by forward mode AD is equal to the cost of computing $w^T J$ by reverse mode AD, where v, w are vectors. This is true in the order of magnitude sense – both computations take time proportional to $\omega(F)$ – but the respective constants may differ widely. It may be pragmatic to estimate "weights" w_1, w_2, with respect to a given AD tool, reflecting the relative costs of forward and reverse modes. It is very easy to introduce weights into algorithm $MNCO$ (§4.3) to heuristically solve a "weighted" problem, $\min w_1\chi_1 + w_2\chi_2$, where χ_1 is the number of row groups (or colors assigned to the rows), and χ_2 is the number of column groups (or colors assigned to the columns). The heuristic $MNCO$ can be changed to address this problem by simply changing the conditional (LB) to:

$$if\ w_1 \cdot \rho(J_R^T) + w_2 \cdot max(\rho(J_C), nnz(r)) \ < w_1 \ \cdot \rho(J_C) + w_2 \cdot max(\rho(J_R^T), nnz(c)).$$

Different weights produce different allocations of work between forward and reverse modes, skewed to reflect the relative costs. For example, consider a 50-by-50 grid matrix with $DENS = 1$, (see §5), and let us vary the relative weighting of forward versus reverse mode: $w_1 = 0 : .25 : 1$, and $w_2 = 1 - w_1$. The results of our weighted bi-coloring approach are given in Table 8.

w_1	χ_1	χ_2
0	50	0
0.25	16	7
0.5	10	10
0.75	7	16
1.0	0	50

Table 8: Weighted problem results

Finally, we note that the bi-coloring ideas can sometimes be used to efficiently determine relatively dense Jacobian matrices provided structural information is known about the function F. For example, suppose $F : \Re^n \rightarrow \Re^m$ is a partially separable function, $F = F_1 + F_2 + \cdots + F_t$, where $F_i : \Re^n \rightarrow \Re^m$, $i = 1 : t$, and each component function F_i typically depends on only a few components of x. Clearly each Jacobian function J_i is sparse while the summation, $J = \sum_{i=1}^t J_i$, may or may not be sparse depending on the sparsity patterns. However, if we define a "stacked" function $\tilde{F}$,

$$\tilde{F}(x) = \begin{pmatrix} F_1(x) \\ F_2(x) \\ \vdots \\ F_t(x) \end{pmatrix}$$

then the Jacobian of $\tilde{F}$ is

$$\tilde{J} = \begin{pmatrix} J_1 \\ J_2 \\ \vdots \\ J_t \end{pmatrix}.$$

Clearly $\tilde{J}$ is sparse and the bi-coloring/AD technique can be applied to $\tilde{J}$, possibly yielding a dramatic decrease in cost. Specifically, if J is dense (a possibility) then the work to compute J without exploiting structure is $n \cdot \omega(F)$ whereas the cost of computing $\tilde{J}$ via bi-coloring/AD is approximately $\chi_c(\mathcal{G}_b(\tilde{J})) \cdot \omega(F)$ flops, where $\chi_c(\mathcal{G}_b(\tilde{J}))$ is the minimum number of colors required for a bipartite cyclic coloring of graph $\mathcal{G}_b(\tilde{J})$. Typically, $\chi_c(\mathcal{G}_b(\tilde{J})) << n$. The idea of applying the bi-coloring/AD technique in a structured way is not restricted to partially separable functions [9].

8 Acknowledgements

We are very grateful to Andreas Griewank and his students Jean Utke and David Juedes for helping us with the use of ADOL-C.

References

[1] B. M. Averick, J. J. Moré, C. H. Bischof, A. Carle, and A. Griewank, *Computing large sparse Jacobian matrices using automatic differentiation*, SIAM Journal on Scientific Computing, 15 (1994), pp. 285–294.

[2] C. H. Bischof, A. Bouaricha, P. M. Khademi, and J. J. Moré, *Computing gradients in large-scale optimization using automatic differentiation*, Tech. Report MCS-P488-0195, MCS, June 1995.

[3] D. Brelaz, *New methods to color the vertices of a graph*, Comm. ACM, 22 (1979), pp. 251–256.

[4] T. F. Coleman and J.-Y. Cai, *The cyclic coloring problem and estimation of sparse Hessian matrices*, SIAM J. Alg. Disc. Meth., 7 (1986), pp. 221–235.

[5] T. F. Coleman, B. S. Garbow, and J. J. Moré, *Software for estimating sparse Jacobian matrices*, ACM Trans. Math. Software, 10 (1984), pp. 329–345.

[6] ——, *Software for estimating sparse Hessian matrices*, ACM Trans. Math. Software, 11 (1985), pp. 363–377.

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

[8] ——, *Estimation of sparse Jacobian matrices and graph coloring problems*, SIAM J. on Numerical Analysis, 20 (1984), pp. 187–209.

[9] T. F. Coleman and A. Verma, *Structure and efficient Jacobian calculation*, Tech. Report in preparation, Computer Science Department, Cornell University, 1995.

[10] A. R. Curtis, M. J. D. Powell, and J. K. Reid, *On the estimation of sparse Jacobian matrices*, J. Inst. Math. Appl., 13 (1974), pp. 117–119.

[11] A. Griewank, *Direct calculation of Newton steps without accumulating Jacobians*, in Large-Scale Numerical Optimization, T. F. Coleman and Y. Li, eds., SIAM, Philadelphia, Penn., 1990, pp. 115–137. Also appeared as Preprint MCS–P132–0290, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., February 1990.

[12] ——, *Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation*, Optimization Methods and Software, 1 (1992), pp. 35–54. Also appeared as Preprint MCS–P228–0491, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., April 1991.

[13] ——, *Some Bounds on the Complexity of Gradients, Jacobians, and Hessians*, in Complexity in Nonlinear Optimization, P. Pardalos, ed., World Scientific Publishers, 1993, pp. 128–161.

[14] A. Griewank, D. Juedes, J. Srinivasan, and C. Tyner, *ADOL-C, a package for the automatic differentiation of algorithms written in C/C++*, ACM Trans. Math. Software, (to appear). Also appeared as Preprint MCS–P180–1190, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., November 1990.

[15] A. K. M. Hossain and T. Steihaug, *Computing a sparse Jacobian matrix by rows and columns*, Tech. Report 109, Department of Informatics, University of Bergen, June 1995.

[16] M. J. D. Powell and P. L. Toint, *On the estimation of sparse Hessian matrices*, SIAM J. Numer. Anal., 16 (1979), pp. 1060–1074.