Computing sparse derivatives and consecutive zeros problem

B V Ravi Chandra\textsuperscript{1} and Shahadat Hossain\textsuperscript{2}

\textsuperscript{1} Birla Institute of Technology and Sciences (BITS), Pilani, India
\textsuperscript{2} Department of Mathematics and Computer Science, University of Lethbridge, Alberta T1K 3M4, Canada
E-mail: \textsuperscript{1} ravichandra363@gmail.com, \textsuperscript{2} shahadat.hossain@uleth.ca

Abstract. We describe a substitution based sparse Jacobian matrix determination method using algorithmic differentiation. Utilizing the a priori known sparsity pattern, a compression scheme is determined using graph coloring. The “compressed pattern” of the Jacobian matrix is then reordered into a form suitable for computation by substitution. We show that the column reordering of the compressed pattern matrix (so as to align the zero entries into consecutive locations in each row) can be viewed as a variant of traveling salesman problem. Preliminary computational results show that on the test problems the performance of nearest-neighbor type heuristic algorithms is highly encouraging.

1. Introduction

We consider the problem of computation or estimation of the sparse Jacobian matrix $J \equiv \left\{ \frac{\partial f_i}{\partial x_j} \right\}$, $1 \leq i \leq m, 1 \leq j \leq n$ of a mapping $f : \mathbb{R}^n \mapsto \mathbb{R}^m$. Mathematical derivatives constitute an important ingredient in important algorithms e.g., Newton’s method in nonlinear optimization. In this paper we assume that the matrix has a priori known fixed sparsity pattern in the domain of interest. Furthermore, for computational efficiency (there are many common and computationally expensive subexpressions in the vector function which should be evaluated only once) and/or because the function is only available as a subroutine call, we compute the whole column of $J$ or $JT$, rather than computing each nonzero unknown separately.

A group of columns in which no two columns have nonzero elements in the same row position is known as structurally orthogonal \cite{3}. If columns $j$ and $l$ are structurally orthogonal, then for each row index $i$, at most one of $J(i, j)$ and $J(i, l)$ can be nonzero. In general, $\sum_j J(i, j) = J(i, l)$ for some $l$ ($l$ will depend on $i$) where the sum is taken over a group of structurally orthogonal columns. An estimate of the nonzero elements in the group can be obtained in

$$\frac{\partial f(x + ts)}{\partial t} \bigg|_{t=0} = f'(x)s = As \approx \frac{1}{\epsilon} [f(x + \epsilon s) - f(x)] \equiv b,$$

with a forward difference (one extra function evaluation) \cite{3} where $b$ is the finite difference approximation and $s = \sum_j e_j$ with $e_j$ denoting the $j$th coordinate vector. Using the forward automatic differentiation \cite{5} the unknown elements in the group are obtained as the product $b = f'(x)s$ accurate within working accuracy. Unfortunately, finding a structurally orthogonal
column partition with minimum number of column groups and equivalently coloring the vertices of the column intersection graph is NP-hard [2, 8]. Therefore, many computationally effective algorithms employ heuristic techniques to find a “good” partition. Vector \( b \) in (1) holds the nonzero unknowns in a orthogonal group of columns. If the columns of \( A \) are partitioned into \( p \) orthogonal groups, (1) takes the form,

\[
b_j \equiv A s_j, \quad j = 1, \ldots, p \text{ or } B \equiv AS,
\]

where the \( j \)th column of \( S \), \( s_j \), is defined by the column indices in \( j \)th orthogonal group, \( j = 1, 2, \ldots, p \). It is clear that since the location of nonzero unknowns of \( A \) are assumed to be known, these values can simply be read from the compressed matrix \( B \). Consider the matrix

\[
A = \begin{pmatrix}
0 & a_{12} & 0 & a_{14} & a_{15} & a_{16} \\
a_{21} & a_{22} & a_{23} & 0 & 0 & a_{26} \\
a_{31} & 0 & a_{33} & a_{34} & a_{35} & 0
\end{pmatrix},
\]

where no two columns are structurally orthogonal. Therefore, the seed matrix \( S \) is simply the \( 6 \times 6 \) identity matrix; the compressed matrix \( B \) will have 6 columns, and hence no savings in AD or FD computations can be realized. Now, with the seed matrix:

\[
S = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}, \quad \text{we have}, \quad B = AS = \begin{pmatrix}
a_{12} & a_{12} & a_{14} & a_{14} & a_{15} & a_{15} + a_{16} \\
a_{21} + a_{22} & a_{22} + a_{23} & a_{23} & 0 & a_{26} \\
a_{31} & a_{33} + a_{34} & a_{34} + a_{35} & a_{35} & a_{35} & a_{35}
\end{pmatrix}.
\]

We can recover the nonzero unknowns by solving a linear system of equations associated with each row. For example, for row 2, we obtain (after removing zero column from the matrix at the left and zero entry from vector \( b \))

\[
x^T \begin{pmatrix}
1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix} = \begin{pmatrix}
a_{21} + a_{22} & a_{22} + a_{23} & a_{23} & a_{26}
\end{pmatrix}, \quad (2)
\]

which is a triangular linear system and can be solved cheaply (i.e., without multiplication or division) for the nonzero unknowns. Hossain and Steihaug [6] show that if each row of matrix \( A \) contains at least \( d \) consecutive zero entries, then a seed matrix \( S \) that results triangular linear systems for each row of \( A \) as discussed above, can be constructed with only \( n - d \) columns. However, reordering the columns so as to maximize \( d \) is NP-hard [6]. In the next section we present a column reordering heuristic and computational test results.

2. Column reordering

The example matrix \( A \) in the preceding section contains two zero entries in each row, and except for row 2, the zeros are not consecutive. However, applying the permutation \( \Pi = (1, 3, 5, 4, 6, 2) \) to the columns, the zero entries in each row of the resulting matrix \( A_\Pi = AP \) where the \( j \)th column of \( P \) is the \( \Pi(j) \)th coordinate vector, become consecutive.

**Consecutive Zeros Problem (CZP).** Given the sparsity pattern of a matrix \( A \in \mathbb{R}^{m \times n} \) find a permutation \( \Pi \) of the columns of \( A \) that maximizes \( d = \min_i \{|\rho_i|; \rho_i \text{ is the maximal sequence of consecutive zero entries in row } i \text{ of } A_\Pi\}, i = 1, 2, \ldots, n. \)

The central idea of our heuristic method is to represent the matrix ordering problem as a variant of traveling salesman problem (TSP). For every pair of columns \( j, l \) of matrix \( A \), we define

\[
c_{jl} = \sum_{i=1}^{n} A(i, j) \odot A(i, l),
\]

where

\[
A(i, j) = \begin{cases}
1 & \text{if } j \in \Pi(i) \text{ and } l \in \Pi(i)
0 & \text{otherwise}
\end{cases}
\]

and \( \odot \) denotes the Hadamard product.
where,
\[
A(i, j) \odot A(i, l) = \begin{cases} 
0, & \text{if } (A(i, j) = A(i, l) = 0) \text{ or } (A(i, j) \neq 0 \text{ and } A(i, l) \neq 0) \\
1, & \text{otherwise}
\end{cases}
\]

Our objective is to find a permutation \( \Pi \) of the columns that minimizes the cost
\[
C(A_{\Pi}) = \sum_{j=1}^{n-1} c_{\Pi(j)\Pi(j+1)}.
\]

For the example matrix from the preceding section, we have \( C(A) = 8 \) and
\( C(A_{\Pi}) = 4 \). The best approximation guarantee of 1.5 for a polynomial time heuristic for TSP
is due to Christofides [1] and can be implemented in \( O(n^3) \). For the CZP we have implemented
a greedy heuristic “nearest-neighbor (NN)” and an improvement heuristic “2-OPT”. In NN
we start at an arbitrary column as the first column in the permutation ordering i.e., \( \Pi(1) \) is
the column selected. At each iterative step \( j > 1 \), we choose column \( l \) such that \( c_{\Pi(j)l} \) is
minimum and set \( \Pi(j+1) = l \). This process is continued until all the columns are ordered.
In our implementation of the 2-OPT improvement heuristic the input is the permuted column
indices obtained from the NN heuristic. At each iterative step, a pair of column indices \( j, l \) are
chosen arbitrarily. A new permutation is created by exchanging their position in the ordering
and its cost is computed. If the new cost is an improvement over the current best, the current
best is updated. Otherwise, no exchange is made. This step is repeated for a user-defined
maximum number of iterations.

### Table 1. Nearest-neighbor heuristic test results

| Name       | m  | n  | \( z_{\min} \) | \( d_1 \) | \( d_2 \) |
|------------|----|----|----------------|--------|--------|
| abb313.mtx | 313| 10 | 4              | 2      | 2      |
| ash219.mtx | 219| 4  | 2              | 1      | 1      |
| ash331.mtx | 331| 6  | 4              | 2      | 2      |
| ash608.mtx | 608| 6  | 4              | 2      | 2      |
| ash958.mtx | 958| 6  | 4              | 2      | 2      |
| will199.mtx| 199| 7  | 1              | 1      | 1      |
| af23560.mtx| 23560| 41 | 20             | 3      | 3      |
| cage11.mtx | 39082| 62 | 31             | 4      | 4      |
| cage12.mtx | 130228| 68 | 35             | 4      | 4      |
| e40r0100.mtx| 17281| 69 | 7              | 1      | 1      |
| fs.541.1.mtx| 541| 13 | 2              | 1      | 1      |
| fs.541.2.mtx| 541| 13 | 2              | 1      | 1      |
| fs.541.3.mtx| 541| 13 | 2              | 1      | 1      |
| impcol_b.mtx| 59 | 11 | 4              | 2      | 2      |
| impcol_d.mtx| 425| 11 | 1              | 1      | 1      |
| impcol_e.mtx| 225| 21 | 9              | 4      | 4      |
| lp_ken_11.mtx| 14694| 125| 3              | 2      | 2      |
| lp_ken_13.mtx| 28632| 171| 1              | 1      | 1      |
| lp_mnaro_r7.mtx| 3136| 83 | 35             | 3      | 3      |
| lund_a.mtx | 147| 13 | 1              | 1      | 1      |
| lund_b.mtx | 147| 13 | 1              | 1      | 1      |

3. Computational experiments

Table 1 displays test results for NN and 2-OPT algorithms on matrix instances drawn from
University of Florida Test Collection [4]; \( m, n, z_{\min}, d_1, d_2 \) denote, respectively, number of rows,
number of columns, minimum number of zeros in any row, value of $d$ (as in CZP) with NN, value of $d$ (as in CZP) with 2-OPT. The pattern matrices in the test suite have been pre-compressed by using structurally orthogonal partitions (or vertex coloring); for 2-OPT algorithm we perform 200 exchange iterations. For the studied test instances, 2-OPT does not yield any improvement in $d$. In fact, using an exhaustive search (by testing all permutations for smaller problems indicated in boldface) we have verified that the NN heuristic yields optimum $d$. For larger problems, we set a time limit of one hour and exhaustive search provided no improvement over NN heuristic. We observe that for a few of the test instances $z_{\text{min}}$ is significantly larger than the value of $d$ computed by the NN heuristic; for test instances with $z_{\text{min}} > 2$, except for problem lp_ken_11.mtx, the savings (with respect to extra function evaluation in FD or AD forward evaluation) can be as high as 50%. The test instances for which the optimality of $d$ can be verified, we get about 19% savings, on average, in computational cost.

4. Concluding remarks
In this paper we have proposed a formulation for the consecutive zeros problem arising in the computation of sparse derivative matrices. The test results indicate savings in computational cost when the Jacobian matrix to be determined is preprocessed with structurally orthogonal partitioning of columns. In iterative schemes e.g., Newton’s method for solving nonlinear system of equations, the most expensive computation at each iteration is the evaluation of the Jacobian matrix. For large-scale problems the overall computational time is greatly reduced by savings in Jacobian evaluation. Our test results indicate that the savings can be as large as 67% over structurally orthogonal column partitioning alone. However, for certain test instances, elimination methods as outlined in [7, 9] can be employed to fully exploit the available sparsity in the matrix.

Acknowledgments
B V Ravi Chandra was supported by a MITACS Globalink Summer Fellowship and Shahadat Hossain’s work was partially supported by a NSERC Discovery Grant (individual).

References
[1] Christofides N 1976 Worst-case analysis of a new heuristic for the travelling salesman problem Management Sciences Research Report 388, CMU
[2] Coleman T. F and J. J. Moré J. J 1983 Estimation of sparse Jacobian matrices and graph coloring problems SIAM J. Numer. Anal., 20(1):187–209
[3] Curtis A. R, Powell M. J. D, and Reid J. K 1974 On the estimation of sparse Jacobian matrices J. Inst. Math. Appl., 13:117–119
[4] Davis T. A and Hu Y The University of Florida sparse matrix collection ACM Trans. Math. Softw. (to appear)
[5] Griewank, A and Walther, A 2008 Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, 2nd edn. No. 105 in Other Titles in Applied Mathematics, SIAM, Philadelphia, PA
[6] Hossain S and Steihaug T 2002 Sparsity issues in the computation of jacobian matrices In Mora T, editor, Proceedings of the International Symposium on Symbolic and Algebraic Computing (ISSAC), pages 123–130, New York, ACM
[7] Hossain S and Steihaug T 2006 Computing sparse Jacobian matrices optimally In Bcker M, Corliss G, Naumann U, Hovland P, Norris B, Barth T J, Griebel M, Keyes D E, Nieminen R M, Roose D, and Schlick T, editors, Automatic Differentiation: Applications, Theory, and Implementations, volume 50 of Lecture Notes in Computational Science and Engineering, pages 77–87. Springer Berlin Heidelberg
[8] Hossain S and Steihaug T 2012 Optimal direct determination of sparse Jacobian matrices Opt. Meth. and Soft. doi: 10.1080/10556788.2012.693927
[9] Newsam G N and Ramsdell J D. 1983 Estimation of sparse Jacobian matrices. SIAM J. Alg. Disc. Meth., 4(3):404–417