

# Pascal Seeding and the Efficient Calculation of Sparse Jacobian Matrices\*

(Extended Abstract)

Shahadat Hossain<sup>†</sup>      Trond Steihaug<sup>‡</sup>

March 29, 2004

## Abstract

An elimination procedure based on Schur complement calculation is proposed for restoring the nonzero elements of a sparse Jacobian matrix. We provide a sparsity usage parameter that can be specified by users to tune the cost of matrix-vector product calculation and the cost of restoration of the nonzero entries. We employ Pascal seeding to compress the Jacobian matrix. Preliminary numerical test results are promising.

**Keywords** Sparse Jacobian Matrices, Indirect Methods, Partition, Pascal Seeding

To determine sparse Jacobian matrices efficiently it is necessary to exploit information such as sparsity and other special structure such as symmetry. Given “seed” matrix  $S \in R^{p \times q}$ , the Jacobian matrix  $A \in R^{m \times p}$  can be “compressed” by computing the product  $AS$  using AD forward mode. The nonzero entries of matrix  $A$  can be recovered (i.e. restored) by solving for them in the linear system of equations

$$AS = B$$

where  $B$  is the (column) compressed Jacobian matrix obtained via AD forward mode. Direct methods allow the nonzero entries to be “read off” the compressed Jacobian. However, optimal direct determination of the Jacobian matrix is hard a problem [7]. On the other hand, with certain class of seed matrices the nonzero entries of  $A$  can be obtained indirectly via substitution or elimination [3]. While indirect methods in general are more efficient than direct methods in that they typically require fewer matrix-vector products, numerical precision of

---

\*This research is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Norwegian Research Council (NFR).

<sup>†</sup>Department of Mathematics and Computer Science, University of Lethbridge, Canada ([shahadat.hossain@uleth.ca](mailto:shahadat.hossain@uleth.ca))

<sup>‡</sup>Department of Informatics, University of Bergen, Norway ([trond.steihaug@ii.uib.no](mailto:trond.steihaug@ii.uib.no))

the computed quantities may suffer due to round-off errors. In the elimination proposal of Newsam and Ramsdell [8] two classes of seed matrices have been considered: the Vandermonde and the Chebyshev-Vandermonde. While the Vandermonde seeding allows efficient solution of the reduced linear system for the nonzero entries in each row, the poor numerical conditioning makes this method impractical except for when the number of nonzero entries in each row is very small. Geitner, Utke, and Griewank's proposal [1] uses graph coloring to improve the numerical conditioning of the computed entries with Newsam and Ramsdall approach. More recently Griewank and Verma [4, 5] and Hossain and Steihaug [6] suggested elimination schemes that employ new classes of seed matrices. With the Pascal seeding [5, 6] the numerical accuracy of the computed quantities has been found to be very favorable compared with Vandermonde systems. As well, the resulting linear systems can be solved for the unknowns efficiently by utilizing the special structure of the Pascal seed matrix. In the present work, we propose an extension to the elimination scheme that retains the superior numerical conditioning while economizing computational effort in the reduced system solve.

We use MATLAB [9] matrix notation as in Golub and Van Loan [2]. Let  $v$  and  $z$  be vectors of column indices of nonzero and zero elements, respectively, of some row  $r$  of  $A$  and let  $\rho$  denote the number of nonzero elements in row  $r$ . Suppose  $S \in R^{p \times q}$  be a Pascal seed matrix and assume that the compressed Jacobian  $B = AS$  has been computed. Define  $Z \in R^{d \times p}$ ,  $d = (p - q)$ ,  $q \geq \rho$

$$Z(i, :) = e_{z(i)}^T, \quad i = 1, \dots, d$$

where  $e_k$  is the  $k$ th coordinate vector. Now consider the matrix

$$W = \begin{pmatrix} S^T \\ Z \end{pmatrix},$$

and let

$$\begin{aligned} A(i, v) &= (\alpha_1 \quad \cdots \quad \alpha_\rho) = \alpha^T, \alpha \in R^\rho, \\ B(i, :) &= (\beta_1 \quad \cdots \quad \beta_p) = \beta^T, \beta \in R^p. \end{aligned}$$

Then,

$$Wx = b \tag{1}$$

with  $x = (\alpha \mathbf{0})$  and  $b = (\beta \mathbf{0})$  with appropriately dimensioned zero vector  $\mathbf{0}$ . Let the matrix  $W$  be partitioned as  $2 \times 2$  blocks so that the Equation (1) can be written as:

$$\begin{pmatrix} S_1 & S_2 \\ Z_1 & Z_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \tag{2}$$

where  $S_1 = S(1 : q, 1 : q)^T$  is the upper triangular part of the Pascal seed  $S^T$  and the vectors  $x$  and  $b$  are partitioned accordingly. Let  $0_j$  denote the zero vector in  $R^j$ . Then column  $i$  in the seed matrix  $S$  is (row  $i$  in  $S^T$ )

$$[0_{i-1} \quad u \quad 0_{p-d-i}]$$

where component  $j$  in  $u \in R^{d+1}$  is the binomial coefficient  $\binom{d}{j-1}$ . Equation (2) can be solved for  $x$  as below.

1. Compute:  $-Z_1 S_1^{-1} S_2 + Z_2$
2. Solve for  $x_2$ :  $(-Z_1 S_1^{-1} S_2 + Z_2) x_2 = -Z_1 S_1^{-1} \beta$
3. Solve for  $x_1$ :  $S_1 x_1 = \beta - S_2 x_2$

In step (1) computing the product  $Z_1 S_1^{-1}$  correspond to simply picking out the rows in the inverse of  $S_1$  corresponding to the known zeros and  $Z_1 S_1^{-1} S_2$  is computed as  $d$  matrix-vector products.  $Z_2$  is 0 – 1 matrix. Hence we have to solve a  $d \times d$  system. An important observation here is to note the special properties of block  $S_1$ .  $S_1$  is upper triangular and the nonzero entries are the binomial coefficients:

$$S_1(i, j) = \begin{cases} \binom{d}{j-i} & \text{if } j \geq i, |j-i| \leq d, 1 \leq i, j \leq d+2. \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

The structure of matrix  $S_1$  is shown below.

$$S_1 = \begin{pmatrix} \binom{d}{0} & \binom{d}{1} & \binom{d}{2} & \cdots & \binom{d}{d} & 0 \\ 0 & \binom{d}{0} & \binom{d}{1} & \cdots & \binom{d}{d-1} & \binom{d}{d} \\ 0 & 0 & \binom{d}{0} & \cdots & \binom{d}{d-2} & \binom{d}{d-1} \\ \vdots & & & & & \\ 0 & 0 & 0 & \cdots & 0 & \binom{d}{0} \end{pmatrix}.$$

Then  $S_1^{-1}$  is upper triangular and is given by

$$S_1^{-1}(i, j) = \begin{cases} (-1)^{j-i} \binom{d+j-i-1}{d-1} & \text{if } j \geq i \\ 0 & \text{otherwise} \end{cases}, 1 \leq i, j \leq d+2. \quad (4)$$

Figure (1) shows the Schur-complement elimination matrix  $W$  with  $p = 8$  and  $q = 5$ . Vector  $z$  shows that columns 1, 4, and 5 contain zero in some row  $r$  of the Jacobian matrix. The remaining  $q = 5$  elements are nonzero which are to be determined. We must note that  $\rho \leq q \leq p$  is an important user supplied parameter. With  $q = \rho$  the number of matrix-vector products needed is minimal but with a corresponding increase in the computational cost for restoring the nonzero elements, while  $q = 0$  results in direct determination of the nonzero entries.

Figure 1: Schur complement method with Pascal seeding when  $p = 8$  and  $q = 5$ .

$$W = \begin{pmatrix} 1 & 3 & 3 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 3 & 3 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 3 & 3 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}, S_1 = \begin{pmatrix} 1 & 3 & 3 & 1 & 0 \\ 0 & 1 & 3 & 3 & 1 \\ 0 & 0 & 1 & 3 & 3 \\ 0 & 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, z = ( 1 \ 4 \ 5 )$$

Using (4), the inverse of  $S_1$

$$S_1^{-1} = \begin{pmatrix} 1 & -3 & 6 & -10 & 15 \\ 0 & 1 & -3 & 6 & -10 \\ 0 & 0 & 1 & -3 & 6 \\ 0 & 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

We have no rounding errors in computing the Schur complement since it is all integer computations. An estimate of the numerical conditioning of  $S_1$  in  $L_1$  norm can be given as

$$\kappa_1 = \|S_1\|_1 \|S_1^{-1}\|_1 = \sum_{i=0}^d \binom{d}{i} \sum_{i=0}^d \binom{d+i-1}{d-1} = 2^d \binom{2d+1}{d}$$

Conditioning of the step 3 is the worst possible [6]. This poor conditioning, however, does not seem to cause too much problem numerically. Results from preliminary numerical testing show good performance. More extensive numerical testing is currently being performed.

## References

- [1] U. Geitner, J. Utke, and A. Griewank. Automatic computation of sparse Jacobians by applying the method of Newsam and Ramsdell. In M. Berz, C. Bischof, G. Corliss, and A. Griewank, editors, *Computational Differentiation: Techniques Applications, and Tools*, pages 161–172. SIAM, Philadelphia, Penn., 1996.

- [2] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.
- [3] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, Penn., 2000.
- [4] A. Griewank and A. Verma. The Newsam-Ramsdell-Lagrange Method for Calculating Sparse Jacobians. Technical Report Preprint IOKOMO-07-2001, TU Dresden, 2001.
- [5] A. Griewank and A. Verma. The Newsam-Ramsdell-Lagrange Method for Calculating Sparse Jacobians, 2002. SIAM Conference on Optimization.
- [6] S. Hossain and T. Steihaug. Sparsity Issues in the Computation of Jacobian Matrices. In T. Mora, editor, *Proceedings of the International Symposium on Symbolic and Algebraic Computing (ISSAC)*, pages 123–130, New York, 2002. ACM.
- [7] S. Hossain and T. Steihaug. Optimal Direct Determination of Sparse Jacobian Matrices. Technical Report 254, Department of Informatics, University of Bergen, Norway, October 2003.
- [8] G. N. Newsam and J. D. Ramsdell. Estimation of sparse Jacobian matrices. *SIAM J. Alg. Disc. Meth.*, 4(3):404–417, 1983.
- [9] The MathWorks. Matlab 5.3.1, 2000. See [www.mathworks.com/products/matlab](http://www.mathworks.com/products/matlab).