NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2001; **8**:1–6

Reversing the row order for the row-by-row frontal method

J. K. Reid*,† and J. A. Scott

Computational Science and Engineering Department, Atlas Centre, Rutherford Appleton Laboratory, Oxon, OX11 0QX, U.K.

SUMMARY

The efficiency of the row-by-row frontal method for the solution of unsymmetric sparse linear systems of equations is dependent on the row ordering used. Numerical experience has shown us that it can be advantageous to reverse a given row ordering. We present two results on invariances under the reversal of the ordering and use real applications to illustrate the variations that can take place upon row reversal. Copyright © 2001 John Wiley & Sons, Ltd.

KEY WORDS: ordering rows; frontal method; sparse unsymmetric matrices

1. INTRODUCTION

The row-by-row frontal method (see, for example, Duff, Erisman and Reid [1, Section 10.6]) solves a large sparse unsymmetric set of n linear equations by Gaussian elimination with the help of a full rectangular matrix that is held in memory and is known as the *frontal matrix*. The size of the frontal matrix varies during the elimination. Rows are *assembled* (added) into the frontal matrix one by one. Whenever a column becomes *fully summed*, that is, the last row in which it has an entry is assembled, a pivot is chosen in the column and is used to eliminate the column and the row containing the pivot. The eliminated row and column are stored for use in the back-substitution or in the solution of further systems of equations.

Since an elimination can only take place after a column becomes fully summed, the order in which the rows are assembled will determine both how long a variable remains in the front and the order in which the variables are eliminated (apart from the order among columns that become fully summed at the same assembly step, which has no effect on the computational requirements). For efficiency, in terms of both storage and arithmetic operations, the rows need to be assembled in an order that keeps the size of the frontal matrix as small as possible. Scott [2] has considered a number of strategies for determining such an ordering and has found that the results of using the different orderings on practical problems can vary enormously. While developing row ordering software for HSL 2000 [3], Scott tried experimenting with reversing

*Correspondence to: J. Reid, Computational Science and Engineering Department, Atlas Centre, Rutherford Appleton Laboratory, Oxon, OX11 0QX, U.K.
†E-mail: j.k.Reid@rl.ac.uk

Received 4 June 1999
Accepted 30 March 2000

any given row order. She found that this makes no difference to some of the usual measures of the quality of the ordering but it can make enormous changes to others.

These numerical results motivated us to prove the results on invariances under the reversal of the ordering that we present in Section 2. Theorem 2.1 is straightforward and unsurprising. Theorem 2.2 took us by surprise. We began by seeking a counter-example by computer. Finding none, we sought a proof, the honed version of which we present here. In Section 3, we illustrate the variations that can take place in real applications.

2. TWO THEOREMS ON INVARIANCE

We follow Scott [2] and use the terms *row frontsize* and *column frontsize* for the number of rows and columns in the front. Our invariance results pertain to the column frontsizes.

The row and column frontsizes *after assembly* refer to the frontsizes after a row has been added to the front. When a row is added, more than one column may become fully summed; for each such column, a pivot is chosen and an elimination is performed. The row and column frontsizes *before elimination* refer to the frontsizes immediately prior to an elimination. The *frontal matrix size* refers to the product of the row and column frontsizes.

We use the term *forward* row order to refer to the row ordering $1, 2, \ldots, n$ (that is, to the given ordering), while the *reverse* ordering is $n, n-1, \ldots, 1$. We assume that the matrix (in the forward order) is $A = \{a_{ij}\}$ and that a reference to 'row i' refers to the row $a_{i\bullet}$.

Before presenting our results, we introduce a small illustrative example for which the matrix (with row numbers added) in the forward and reverse orders is

$$\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\!\!\begin{pmatrix} x & & & & \\ x & x & x & x & \\ x & x & x & x & \\ & x & & x & \\ & x & & & x \end{pmatrix} \qquad \begin{array}{c} 5 \\ 4 \\ 3 \\ 2 \\ 1 \end{array}\!\!\begin{pmatrix} & x & & & x \\ & x & & x & \\ x & x & x & x & \\ x & x & x & x & \\ x & & & & \end{pmatrix}$$

The frontal matrix sizes after each assembly and elimination operation for the forward and reverse orderings are as follows:

Forward order		Reverse order	
Assemble row 1	1×1	Assemble row 5	1×2
Assemble row 2	2×4	Eliminate column 5	0×1
Assemble row 3	3×4	Assemble row 4	1×2
Eliminate column 3	2×3	Assemble row 3	2×4
Eliminate column 1	1×2	Assemble row 2	3×4
Assemble row 4	2×2	Eliminate column 4	2×3
Eliminate column 4	1×1	Eliminate column 3	1×2
Assemble row 5	2×2	Eliminate column 2	0×1
Eliminate column 5	1×1	Assemble row 1	1×1
Eliminate column 2	0×0	Eliminate column 1	0×0

Thus, the row frontsize, column frontsize, and the frontal matrix size before each of the five eliminations together with their mean values are

before elimination of column	3	1	4	5	2	
row frontsize	3	2	2	2	1	2.0
column frontsize	4	3	2	2	1	2.4
frontal matrix size	12	6	4	4	1	5.4

When the order is reversed, the corresponding statistics are

before elimination of column	5	4	3	2	1	
row frontsize	1	3	2	1	1	1.6
column frontsize	2	4	3	2	1	2.4
frontal matrix size	2	12	6	2	1	4.6

Our first result is about the column frontsize after each assembly. This is important since the size of the frontal matrix after assembly determines how much memory is needed by the frontal solver.

Theorem 2.1. *If a given row order is reversed, the sequence of n column frontsizes after the assemblies is reversed.*

Proof
After assembly of row i, in the forward or reverse row order, the column frontsize is the number of columns with first entry at row i or earlier and last entry at row i or later, that is, the number of columns j for which

$$\min\{k\colon a_{kj} \neq 0\} \leqslant i \quad \text{and} \quad \max\{k\colon a_{kj} \neq 0\} \geqslant i.$$

Corollary 2.1. *The maximum column frontsize is invariant.*

Corollary 2.2. *The mean column frontsize after assembly is invariant.*

Corollary 2.3. *The sum of the column frontsizes after assembly is invariant.*

Note that the sum of the column frontsizes after assembly is equal to the sum of the lifetimes, where the lifetime of a variable is defined to be the number of assembly steps for which the variable is in the front. The lifetime of variable j is also the length of column j, that is, $\max\{k\colon a_{kj} \neq 0\} - \min\{k\colon a_{kj} \neq 0\}$. The sum of the lifetimes is used by Camarda [4] to compare the quality of different row orderings.

Our second result concerns the column frontsize before each elimination. This is important since the row and column frontsizes before an elimination determine how much work is associated with the elimination and how much storage is required for the pivotal column and row.

Theorem 2.2. *If a given row order is reversed, the sequence of n column frontsizes before the eliminations is permuted.*

Proof
At assembly step i in the forward order, the column frontsize increases by one for each column with its first entry in row i, that is, each j such that $\min\{k\colon a_{kj} \neq 0\} = i$. Following this, it decreases by one for each column with its last entry in row i, that is, each j such that $\max\{k\colon a_{kj} \neq 0\} = i$, as that column is eliminated. There are n occasions when it increases by one and n occasions when it decreases by one. These $2n$ events begin and end with the column frontsize of zero. For each increase from $f-1$ to f there is a corresponding later decrease from f to $f-1$, and to establish a one-one correspondence, we take the first such decrease if there is more than one. Each decrease corresponds to an elimination with column frontsize f using the forward row order. Each increase corresponds to an elimination with column frontsize f using the reverse row order. The result follows.

For the forward order on our small example, the sequence of column frontsizes used in this proof is $0, 1, 2, 3, 4, 3, 2, 1, 2, 1, 0$. Note that there are two increases from 1 to 2 and two decreases from 2 to 1; the first increase is taken to correspond with the first decrease and the second increase to correspond with the second decrease.

Corollary 2.4. *The mean column frontsize before elimination is invariant.*

Note that the mean values in Corollaries 2.2 and 2.4 usually differ, while there is only one maximum value. For the forward order on our small example, the mean column frontsizes after assembly and before elimination are 2.6 and 2.4, respectively.

3. NUMERICAL EXPERIENCE

We now present some numerical results that illustrate our theoretical results. All the frontsizes quoted in this section are frontsizes *before elimination*. The test problems are those used by Scott [2]; a brief description of each problem is given in Table I. The problems all arise from real engineering or industrial applications. Our numerical results illustrating the effects of

Table I. The test problems.

Identifier	Order	Number of entries	Description/discipline
bayer04	20545	159082	Chemical process simulation
bayer09	3083	21216	Chemical process simulation
bp1600	1600	4841	Basis matrix from LP problem
extr1	2837	11407	Dynamic simulation problem
gre1107	1107	5664	Simulation studies in computer systems
hydr1	5308	23752	Dynamic simulation problem
lhr07c	7337	156508	Light hydrocarbon recovery
lhr14c	14270	307858	Light hydrocarbon recovery
meg1	2904	58142	Chemical process simulation
onetone2	36057	227628	Harmonic balance method, one-tone
orani678	2529	90158	Economic model
rdist1	4134	94408	Reactive distillation problem
west2021	2021	7353	Chemical engineering

Table II. The maximum (max) and mean row and column frontsizes before elimination for the forward and reverse row ordering.

Identifier	Forward				Reverse			
	Max row	Max col.	Mean row	Mean col.	Max row	Max col.	Mean row	Mean col.
bayer04	349	501	174	275	263	501	105	275
bayer09	83	153	38	62	78	153	26	62
bp1600	242	324	91	142	143	324	32	142
extr1	32	46	15	27	21	46	13	27
gre1107	95	208	55	125	123	208	72	125
hydr1	45	86	22	43	42	86	22	43
lhr07c	169	225	51	111	127	225	66	111
lhr14c	164	366	69	188	255	366	125	188
meg1	700	1150	368	639	517	1150	235	639
onetone2	297	658	198	405	365	658	209	405
orani678	1333	1576	604	771	320	1576	70	771
rdist1	40	81	28	60	49	81	32	60
west2021	38	52	18	29	22	52	12	29

Table III. The maximum and mean frontal matrix size and the sum of lifetimes ($*10^3$) for the forward and reverse row ordering.

Identifier	Forward			Reverse		
	Max frontal matrix size	Mean frontal matrix size	Sum of lifetimes	Max frontal matrix size	Mean frontal matrix size	Sum of lifetimes
bayer04	17365	5341	5708	12387	3340	5708
bayer09	1237	283	194	1193	197	194
bp1600	6607	1833	100	4633	659	100
extr1	147	43	76	84	37	76
gre1107	1925	814	140	2489	1055	140
hydr1	387	104	226	361	104	226
lhr07c	3489	622	858	2331	791	858
lhr14c	5953	1530	2755	8992	2849	2755
meg1	77970	28129	1750	54395	18375	1750
onetone2	19483	8646	14637	23980	9264	14637
orani678	210080	64517	1704	48247	8679	1704
rdist1	275	169	242	397	202	242
west2021	198	57	58	86	36	58

row reversal are presented in Tables II and III. In each case, the matrix is reordered using our new code MC62 [5], with default settings for all the control variables.

From the tables, we see that for some problems, including bp1600 and orani678, it can be extremely advantageous to reverse the row order. For other problems, such as hydr1, reversal has little effect. Note also that reversing the order can reduce the maximum row frontsize while increasing the mean row frontsize or the mean frontal matrix size. This is illustrated by

problem `lhr07c`. In this case, which is the better ordering depends upon whether the prime concern is to reduce main memory requirements (choose the smaller maximum frontal matrix size), to minimize factor storage (choose the smaller mean row frontsize), or to minimize the operation count (choose the smaller mean frontal matrix size). Since the cost of computing the frontsizes for the forward and reverse orders is negligible compared with the cost of using a frontal solver, `MC62` computes a new row ordering and then automatically reverses it and selects the better of the two orderings. By default, `MC62` chooses the ordering for which the mean frontal matrix size is smaller.

REFERENCES

1. Duff IS, Erisman AM, Reid JK. *Direct Methods for Sparse Matrices*. Oxford University Press: England, 1986.
2. Scott JA. A new row ordering strategy for frontal solvers. *Numerical Linear Algebra and Applications*, 1999; **6**:1–23.
3. HSL 2000. *A Collection of Fortran Codes for Large Scale Scientific Computation*. http://www.numerical.rl.ac.uk/hsl.
4. Camarda KV. Ordering strategies for sparse matrices in chemical process simulation. *PhD thesis*, University of Illinois at Urbana-Champaign, 1997.
5. Scott JA. Row ordering for frontal solvers in chemical process engineering. *Computers and Chemical Engineering*, 2000; **24**:1865–1880.

