

Comments on "A Fast Computer Method for Matrix Transposing" and Application to the Solution of Poisson's Equation

ULRICH SCHUMANN

In the above correspondence,¹ an algorithm has been described that allows the transposing of a $2^n \times 2^n$ matrix by reading and writing n times at the most via direct access rows of the matrix and by permuting its data elements. The generalization to arbitrary square matrices has not been described, and the generalization to nonsquare matrices has been reported to be impossible.

The algorithm proposed is very similar to one described and proved completely in [1], which needs only two sequential data sets; it has been developed to transpose arbitrary square and nonsquare matrices. The idea of this algorithm is as follows: The $M \times N$ matrix is assumed to be stored by row on a sequential data set. The number N is the product of its prime factors $P_i, i = 1, 2, \dots, n$. Then n steps are needed to transpose the matrix. In every step ($i = 1, n$) the data elements are read from the data set in the following order: $1, 1 + P_i, 1 + 2P_i, \dots, 1 + MN - P_i; 2, 2 + P_i, \dots, 2 + MN - P_i; \dots; P_i, P_i + P_i, P_i + 2P_i, \dots, M \times N$ and written on the second data set sequentially. In doing this, the first data set must be read P_i times, and the second must be written once. Then the data sets are interchanged and the i th step is finished. After all n steps, the last output data set contains the matrix, now stored by column, and for the total process the data have to be read as often as the sum of the prime factors and written n times; so a total of $n + \sum_{i=1}^n P_i$ transputs (input + output) are needed. In the case of $M = N = 2^n$ this sum results in $3n$. By combining pairs of 2 to "prime number" 4, the algorithm may be refined to reduce this sum to $2.5n$.

This is much less than the $(N + 1)$ transputs needed when transposing the matrix stored on a sequential data set in a direct way. Because the number of transputs approach this bad number if N is a prime number itself, the matrix should be expanded to a larger one with row length N_{opt} in the first step by adding meaningless data. Using the optimum numbers N_{opt} of Table I, the number of transputs are always less than $3 \log_2 N_{opt}$ and equal to $2.5 \log_2 N_{opt}$ in the limit [1].

In Eklundh's algorithm only $2n$ transputs are needed, due to the use of direct access transputs. So his algorithm is superior in this case. But the algorithm described above is more general since it is applicable to nonsquare matrices and arbitrary square matrices. Moreover the permutation described above and presented formally in [1, eq. (1)] may be used to generalize Eklundh's algorithm to arbitrary square matrices.

Besides the applications mentioned in [1] and those for fast Fourier transform (FFT), the algorithm has been used for the direct solution of the Poisson equation $\nabla^2 p = d$ in two and three dimensions over a Cartesian and cylindrical grid with about 10^6 nodes. This method will be described shortly for the two-dimensional case of the discrete Poisson equation.

$$(p_{i-1,j} - 2p_{i,j} + p_{i+1,j})/\Delta x^2 + (p_{i,j-1} - 2p_{i,j} + p_{i,j+1})/\Delta y^2 = d_{i,j}$$

over the Cartesian grid

$$x_{i,j} = (i-1)\Delta x \quad y_{i,j} = (j-1)\Delta y, \quad i = 1, 2, \dots, N; \quad j = 1, 2, \dots, N$$

with Dirichlet, Neumann, or periodical boundary conditions. The solution is found in the following way (mathematically according to Hockney [2]).

Step 1: Expand $d_{i,j}$ for one coordinate (e.g., x) into "eigenfunktionen" $d_{i,j} = \sum_{i=1}^N A_{i,j} \exp(\sqrt{-1} 2\pi (i-1)(j-1)/N)$, and evaluate $A_{i,j}$ by FFT (N times for $j = \text{constant}$).

Step 2: Expand $p_{i,j}$ accordingly: $p_{i,j} = \sum_{i=1}^N B_{i,j} \exp(\sqrt{-1} 2\pi (i-1)(j-1)/N)$; substitution into the above Poisson equation results in N tridiagonal linear equation systems for N unknowns $B_{i,j}$ ($i = \text{constant}$), which may be solved in a direct way very easily.

Step 3: Compute $p_{i,j}$ out of $B_{i,j}$ by N FFT's, each for $j = \text{constant}$. The arrays $d_{i,j}; A_{i,j}; B_{i,j}; p_{i,j}$ ($i = 1, N; j = 1, N$) may overstore one array $S_{i,j}$ successively. For $N > 1000$ the N^2 elements of S cannot be stored in the main storage of today's computers. S must be cut into pages which are swapped by hardware or software between a background storage device and the main storage.

Initially it is suitable to design the pages so that just one column ($j = \text{constant}$) can be stored in one page. Steps 1 and 3 may be executed without problems—for each of the N FFT's one page is needed, and the pages are to be read and written $4N$ times.

But in executing Step 2 for each of the N linear equation solutions, a row ($i = \text{constant}$) of S is needed. This means just one data element out of each page. Therefore, this step requires $2N^2$ transputs of pages and, thus, $4N + 2N^2$ are needed for the complete solution.

This unacceptable high number of page swaps can be decreased dramatically by transposing the matrix S according to [1] between Steps 1, 2 and 3. Then only approximately (exact for $N = 2^n, n$ even)

$8N + 5N \log_2 N$ transputs are required to get the complete solution. Thus, in the case of $N = 1024$, by using the matrix transposing algorithms, the number of page swaps are reduced by a factor of about 35.

TABLE I
OPTIMUM VALUES N_{opt} OF THE ROW LENGTH

1	2	3	4	5
6	7	8	9	10
12	15	16	18	20
21	24	25	27	28
30	32	36	40	45
48	50	54	56	60
64	72	75	80	81
84	90	96	100	108
112	120	125	128	135
144	150	160	162	168
180	192	200	216	225
240	243	256	270	288
300	320	324	336	360
375	384	400	405	432
448	450	480	486	500
512	540	576	600	625
640	648	675	720	729
768	800	810	864	900
960	972	1024	1080	1125
1152	1200	1215	1280	1296
1344	1350	1440	1458	1500
1536	1600	1620	1728	1792
1800	1875	1920	1944	2000
2025	2048	2160	2187	2304
2400	2430	2500	2560	2592
2700	2880	2916	3072	3125
3200	3240	3375	3456	3600
3645	3840	3888	4096	

REFERENCES

- [1] U. Schumann, "Ein Verfahren zum Transponieren grosser, sequentiell gespeicherter Matrizen," *Angew. Inform.*, pp. 213-216, May 1972.
- [2] R. W. Hockney, "A fast direct solution of Poisson's equation using Fourier analysis," *J. Ass. Comput. Mach.*, vol. 12, pp. 95-113, 1965.