

EVALUATION OF ORDERINGS FOR UNSYMMETRIC SPARSE MATRICES*

A. M. ERISMAN†, R. G. GRIMES†, J. G. LEWIS†, W. G. POOLE, JR.‡ AND H. D. SIMON†

Abstract. The computational complexity of obtaining optimal reorderings for performing sparse Gaussian elimination justifies the heuristic nature of all practical reordering algorithms. The concomitant lack of analytic results on heuristic orderings requires that any general comparisons between orderings must be empirical. We report our findings from an extensive investigation of the Markowitz, P^4 and P^5 ordering heuristics applied to a spectrum of sparse matrices. The ordering heuristics differ in overall effectiveness in reducing fill, particularly on certain problem classes, and in their ability to provide stable as well as low fill factorizations. Statistical analyses indicate that certain easily observed, general, structural parameters of the matrices are predictors for the amount of fill observed. Various estimates of the stability of the factorizations were compared for reliability and accuracy. We close with recommendations for use and for further investigation.

Key words. sparse matrices, Gaussian elimination, Markowitz ordering, P^4 algorithm

AMS (MOS) subject classification. 65F05

1. Introduction. It is widely believed that the effectiveness of ordering algorithms for sparse matrices depends in large part on the origins of the problems. Yet, the heuristic nature of these algorithms often does not permit analytic results. Realistic evaluation of heuristic orderings demands the use of a representative set of test problems. This report contains the results of a comparative study on the effectiveness of general purpose algorithms for reordering unsymmetric sparse matrices. The study is empirical and is based upon an expanded version of the Harwell collection of test matrices [7], [9].

It is common folk knowledge that sparse matrices which occur in practice have special features that reflect, in unclear fashion, the scientific origins of the problems. The following quote from [7] indicates this wisdom: "Telling an expert that a matrix has arisen from circuit theory or structural mechanics immediately gives him a considerable feel for its features." While the statement certainly contains a good deal of truth, it is, like most results on sparse orderings, hardly a theorem. It is also irrelevant to the inexpert consumer of sparse matrix software, whose desire is to use a general purpose ordering code to solve the actual problem that has (possibly) arisen in some discipline unknown to the software designer.

The goals of this study were two-fold:

—To study the effectiveness of *orderings* across test problems spanning a wide spectrum of scientific origins;

—To investigate whether any simple measures of problem difficulty can be obtained at less cost than actually ordering the problem.

The first goal differs from those of previous studies, particularly that of Duff and Reid [7], in evaluating orderings rather than codes. The selection of data structures has an undeniable impact on the actual computation and on coding practices, but that was explicitly not of interest in this study. Our purpose was to investigate the validity

* Received by the editors August 25, 1983; accepted for publication (in revised form) May 27, 1986. This paper is a result of research supported by the Air Force Office of Scientific Research (AFSC), under contract F49620-81-C-0072.

† Engineering Technology Applications Division, Boeing Computer Services Company, Seattle, Washington 98124.

‡ Pacific Analysis and Computing Corporation, Bellevue, Washington 98004.

of the folk wisdom, especially in evaluating how different orderings would perform when presented with problems not anticipated by the designers of the heuristics.

The data that are the basis of this study, the algorithms and matrices, are discussed in this introductory section, as are the limitations of this study and the conclusions that can be drawn from it. The second section of this report is a discussion of the symbolic, i.e., nonnumeric, measures and properties of the data. The third section discusses some important numerical properties and behavior. In the concluding sections we summarize the results and conclusions, and indicate further directions suggested by this study.

1.1. Algorithms. In contrast to the rich collection of ordering heuristics available for reordering symmetric matrices, the literature contains few orderings designed for any class of unsymmetric matrices, and much less to serve in a general purpose way. Only one algorithm, the Markowitz reordering [23], is commonly accepted as being general in capability. The P^3 and P^4 algorithms by Hellerman and Rarick [17], [18] are frequently used in linear programming codes, also the origin of the Markowitz ordering. In related work [14] the authors developed a modification of the second of the Hellerman-Rarick orderings, which we have denoted as P^5 . Another ordering was developed by Lin and Mah [20]-[22] to solve chemical engineering problems. All of these are based on no special properties other than sparsity, so they could reasonably be applied to quite general problems. Unfortunately, the only implementation of the Lin and Mah algorithm known to the authors is part of a proprietary chemical engineering code and was unavailable to us. Development of a new implementation was beyond the scope of this study, so this ordering was omitted and so were symmetric orderings. These orderings, applied to the connectivity structure of $A + A^T$, are usually viewed by their designers as being applicable only to problems that are nearly symmetric in structure. It has been considered to be ineffective to apply them to problems in which the structure of $A + A^T$ is not close to that of A . They were, therefore, omitted from this study as being not general in purpose.

In applying these orderings to sparse matrices, we precede the ordering with a preordering that places the sparse matrix in a lower block triangular form in which each diagonal block is irreducible. The factorization of a matrix in this form can be performed by independently factoring each diagonal block and, in addition, without causing fill in the blocks in the strict lower triangle [4], [14]. Consequently, the ordering heuristics and the factorization work can be confined to a series of subproblems of smaller order. Fast algorithms have been developed for this task [2], [5], [6], [29]. In general the block reduction results in smaller time and arithmetic requirements, dramatically so if the block form results in diagonal matrices of small order only. There are counterexamples in which ordering the entire matrix is more effective, but generally block triangular reduction is an accepted practice, questioned only for problems expected to be (essentially) irreducible. Indeed, the essential distinction between the P^3 and P^4 algorithms is that the P^4 algorithm is the P^3 algorithm preceded by a reduction to irreducible block triangular form. (Consequently the P^3 algorithm, per se, was not used in this study.) Section 2 contains further discussion of this preordering and of its effectiveness in reducing the complexity of the reordering work for the problems in our test set.

The heuristic due to Markowitz has the simplest statement of the three orderings used in this study, although it is the most complex to implement. It yields a local minimization of the amount of computational work. At each stage in the factorization row and column interchanges are performed so that the leading diagonal element of

the reduced matrix, the next pivot entry, is any element that minimizes the number of operations required at this step of elimination. The reordering can be also considered as a series of local decisions to reduce the amount of fill that occurs, because the minimal operation choice also minimizes a bound on the number of new fill entries. The computation, however, requires that the sparsity structure of the current reduced matrix be known; the ordering routine must perform a partial symbolic factorization as it proceeds. Dynamic data structures capable of representing the structure of the factored matrix, as opposed to the original matrix, are required.

Arbitrary tie-breaking is a common occurrence in heuristic reorderings. All of the orderings of interest to this study contain decision steps in which essentially arbitrary choices are made, as, for example, which to choose among several entries that equally reduce the bound on fill. In some sense then, our study is a study of particular members of classes of orderings.

For a Markowitz type of ordering we used the MA28 code developed by Duff at Harwell [3], [4]. This code has some features that distinguish it from the original ordering by Markowitz, but it is widely accepted as a general purpose ordering and as a Markowitz ordering. A particularly valuable feature of the MA28 code is its attention to numerical stability. The data requirements of the Markowitz ordering are such that it is relatively easy to perform the numerical factorization directly in conjunction with the ordering. This makes the numerical values as well as the structure of the reduced matrix available to the ordering. The numerical values are used by MA28 to control the stability of the factorization by rejecting the natural Markowitz choice if it would cause too much immediate growth in the elements of the factors of the matrix. In such a case an alternative pivot is chosen that minimizes the bound on fill among all the entries that are acceptable on growth grounds. How much growth is acceptable is specified by the user through a tolerance setting which controls the compromise being made between preserving the sparsity of the problem and maintaining the stability of the factorization.

In the study MA28 was used in a mode as near to pure Markowitz as possible by setting the pivot tolerance to zero. The Markowitz ordering may still be perturbed for numerical reasons, but only when the natural Markowitz choice would be accidentally an exactly zero entry. To evaluate the general effectiveness of MA28 as a numerically stable algorithm, corresponding additional tests were made using a particular nonzero value, one tenth, for the growth tolerance.

The P^4 ordering has a significantly more complicated description. The clearest explanation is found in [14]. This heuristic differs from the Markowitz heuristic in two particularly important ways. First, the heuristic is based on obtaining a particular form for the matrix, as opposed to locally reducing fill. The desired form is essentially a lower triangular matrix, in which a few occasional columns known as spikes stick up above the main diagonal. In such a form fill occurs only in the spike columns. This may be used to create less complicated data structures for the factors than are used in MA28. It is also advantageous for updating factorizations when isolated columns of the matrix are changed, as occurs in linear programming problems [26].

A second significant distinction between the P^4 heuristic and the Markowitz heuristic are the data structure requirements for the ordering. Where the Markowitz ordering requires access to the filled reduced matrix, the P^4 ordering proceeds with a static representation of the original matrix. The computational complexity of the ordering is apparently much less than that of Markowitz; an efficient implementation will be much faster and require less storage than a Markowitz ordering. On the other hand, the simple data structures do not allow knowledge of the numerical values that

will arise in the factored matrix. Accidental zero pivots may occur, as may small pivots, which cause unacceptable growth. Indeed, the heuristic may result in nonaccidental structurally zero pivots on the diagonal of the reordered matrix.

The natural way of avoiding such failures of this algorithm requires interchanging the column in which the unacceptable pivot occurs with some particular spike column, which is ordered later in the matrix. This does not affect in any way the ordering of any other columns, so it is usually incorporated in the numerical factorization, which occurs after the ordering. While this maintains the simplicity of the ordering heuristic, it does not allow the heuristic to interact with the changes that are made in the resulting matrix. The result is that fill may increase significantly. Gill et al. [15] and Saunders [26], [27] indicate that this is often the result for even limited control on element growth, and their results do not include measures to demonstrate that the resulting factorizations were acceptable.

The problem of nonaccidental zero pivots in a P^4 ordering can also be avoided by making minor changes in the heuristic. One particular way in which this can be done is the so-called P^5 algorithm, which results from earlier work of the authors. This algorithm is very similar to the P^4 heuristic in its complexity, but the reordered matrix has more structure than that resulting from either P^4 or Markowitz. Advantage of this structure can be taken to reduce dramatically the storage requirements (effective fill) for the factors of the matrix. (The explicit fill can be shown to be never less than for the P^4 ordering). Details are given in [14] and in § 2. Unfortunately, avoiding structurally zero pivots does not guarantee numerical stability. The requirements for stability appear to conflict with the special structure of the factored matrix. Low fill and numerical stability appear to be in direct conflict with this ordering, as is discussed further in § 3.

Two implementations of a P^4 ordering were available to us, one due to Bisschop, Levy and Meeraus at the World Bank [1] and the other by Murtagh and Saunders at Stanford [24]. These differ in slight respects of tie-breaking, which we judged after some experimentation to be insignificant. The World Bank code was more easily modified to create a P^5 code, and so the results herein are for

- Markowitz—the Harwell MA28 code;
- P^4 —the World Bank P^4 code;
- P^5 —the BCS P^5 code based on the World Bank P^4 code.

1.2. Evaluation program. The goals of this study depend on measuring effectiveness of orderings independent of details of implementation. The variability of implementation was removed by using the same code for the symbolic and numerical factorization of the sparse matrices. The primary measures of effectiveness of orderings were judged to be fill and numerical stability. A program was written to extract these statistics from comparable factorizations based on the Markowitz, P^4 and P^5 orderings. In addition, the program computed a number of statistics on the distribution of nonzeros in the original matrix and on the effect of block triangularization. These included a series of measures of regularity and symmetry in the original matrix, as discussed further in § 2.

To isolate ordering properties from details of factorization implementation, the orderings were extracted from the respective codes as permutation vectors. The actual factorizations were performed then by a common code, the so-called refactor code, MA28B, from the Harwell package. In the case of the Markowitz orderings, with and without pivoting, we were able to use the package directly. We first factored the matrices with MA28A, and then refactored identically the same matrices with MA28B. For the other two orderings the permutation vectors were input to a symbolic factorization

subroutine extracted from the MA28A code. The fill is, of course, independent of the factorization implementation. Storage requirements and numerical details do depend on implementation; ours are at least comparable because the implementation is the same for all orderings.

The common factorization module allowed us to compare the effects of the various orderings in general, but it also imposed some limitations on the study. The most important is that no provision was made to investigate the effect of pivoting on the P^4 heuristic. Part of the design of the study was to investigate the effectiveness of P^4 and P^5 as pure preordering schemes. The program design sufficed for this investigation, but makes further investigation of P^4 more difficult. In addition, whenever a structurally zero pivot resulted from the P^4 heuristic, no symbolic factorization could be performed. Although the frequency of this failure was one statistic to be explored in this study, it occurred so frequently as to reduce the overall value of the statistics on the other orderings. The design also affected the evaluation of the P^5 heuristic, since it provided an explicit factorization, which did not use the highly structured matrix form. No meaningful operation counts on the corresponding implicit factorization were obtained, although fill statistics were. The common factorization routine also prohibited using dense code, perhaps with pivoting, for the dense diagonal blocks produced by P^5 . This lack of pivoting may have contributed to some of the numerical failings of the P^5 ordering.

1.3. Matrices. Fifty-eight unsymmetric matrices of various orders were used to test the orderings. Many of these were collected by Duff and Reid [7]. The others were collected by either Duff or the authors, to supplement the coverage of the earlier collection. For purposes of description and of analysis in this study, the fifty-eight problems were classified into six groups. These were

—Chemical Engineering Plant models (sixteen matrices, of order from 59 to 2021; eleven from Prof. A. Westerberg at the University of Pittsburgh; five matrices from Imperial College);

—Linear Programming Bases (sixteen matrices, matrices 12 to 27 in [7], supplied by Michael Saunders);

—Partial Differential Equation models on grids (seven problems, of order 80 to 3564; three from Dr. P. Saylor at the University of Illinois; three 3-D steam models of petroleum reservoirs; one from J. P. Whelan at Philips, Ltd.);

—Simulation matrices with poor numerical behaviors (eleven matrices of order 115 to 1107; four are from atmospheric pollution studies, matrices 33–36 of [7]; seven are computer science simulations from Francois Cachard, Grenoble);

—Miscellaneous (six matrices of order 32 to 199; matrices 4 to 8 and 11 from [7]);

—Optimal power flow problems (two matrices of order 4929 from Rob Burchett, General Electric Company).

Further details on these matrices, including order and initial nonzero density figures, can be found in Table 1.1.

It is clear that these problems cannot stand, in any democratic or statistical sense, as representatives of the universe of unsymmetric sparse matrices. Readers who find this collection unrepresentative of their interests are urged to help correct the situation by contributing to the ongoing effort by Duff and the authors to provide better bases for such studies as this one [9]. This collection does, however, span a wide enough spectrum of problem types to illustrate rather wide differences in behavior among the various orderings across the various classes of problems. The limitations of this test set for testing the dependence of problem complexity on its origin are illustrated by

TABLE 1.1
Test matrices.

Origin	Order	No. of nonzeros
1) Chemical Engineering Plant Models		
from Prof. A. Westerberg, University of Pittsburgh		
Cavett problem	67	294
Rigorous flash unit	132	414
Evaporator/condenser/compressor	156	371
Redlich-Krong Equations of State	167	507
Multiply-feed column	381	2157
Staged column with 8 stages	479	1910
Flash unit with recycling	497	1727
Staged column with 16 stages	655	2854
Staged column with 7 stages	989	3537
Staged column with 11 stages	1105	5445
Staged column with 15 stages	2021	7353
from Imperial College		
Cavett's process	59	312
Ethylene plant	137	411
Heat exchange network	207	572
Hydrocarbon separation	225	1308
Nitric acid plant	425	1339
2) Linear Programming Bases		
from Michael Saunders, Stanford University		
Stair (after 0 iterations in simplex method)	663	1687
(after 200 iterations)	663	1726
(after 400 iterations)	663	1712
Shell (after 0 iterations)	363	2454
(after 200 iterations)	363	3068
(after 400 iterations)	363	3157
(after 600 iterations)	363	3279
BP (after 0 iterations)	822	3276
(after 200 iterations)	822	3807
(after 400 iterations)	822	4028
(after 600 iterations)	822	4172
(after 800 iterations)	822	4534
(after 1000 iterations)	822	4661
(after 1200 iterations)	822	4726
(after 1400 iterations)	822	4790
(after 1600 iterations)	822	4841
3) Simulation Matrices (with poor numerical behavior)		
from Francois Cachard, Grenoble		
Simulation studies in computer science.	115	421
Two of the matrices have identical structure, but differ in their numerical values.	185	1005
	216	876
	216	876
	343	1435
	512	2192
	1107	5664
from A. R. Curtis, Harwell		
Matrices of identical structure and different conditioning, from atmospheric pollution studies	541	4285
involving chemical kinetics and 2D transport,	541	4285
from the FACSIMILE stiff ODE package.	541	4285

TABLE 1.1 (continued)

Origin	Order	No. of nonzeros
4) Partial Differential Equations on Grids from Paul Saylor, Univ. of Illinois		
14 by 17 2D grid	238	1128
10 by 10 by 10 3D grid	1000	3750
33 by 6 by 18 3D grid	3564	22316
collected by Roger G. Grimes, from thermodynamic simulations of oil reservoirs		
4 by 4 by 5 grid, 3 degrees of freedom	240	3762
5 by 5 by 6 grid, 4 degrees of freedom	600	13760
20 point 1D grid, 4 degrees of freedom per element	80	928
J. P. Whelan, Philips Ltd.	991	6027
5) Miscellaneous		
Matrix advertising 1971 IBM sparse matrix conference	32	126
Biochemical ODE Jacobian (A. R. Curtis)	54	291
Jacobian from emitter-follower circuit (R. Willoughby)	57	281
Statistical application (M. Gentleman)	113	655
Jacobian from laser ODE system (A. R. Curtis)	130	1282
Stress analysis matrix (R. Willoughby)	199	701
6) Optimal Power Flow from R. Burchett, General Electric		
Western utility systembasis 0	4929	33185
Western utility systembasis 1	4929	33111

the fact that we originally broke the "chemical engineering" matrices into two classes, because of their different effects on two of the ordering algorithms. The behavior on the third heuristic was quite similar in both groups, an argument that they should not be regarded separately. Similarly the miscellaneous group includes three small Jacobians from ordinary differential equations, whose numerical properties differ greatly from the larger set of simulation problems. The optimal power flow problems were obtained on recommendation of Michael Saunders only after the formal part of the study had been completed. The very distinct behavior of one of the ordering heuristics on these matrices warranted their belated inclusion in this paper. Thus they are listed as a separate group and are not included in our statistical analysis in § 2.

As a test set these matrices do provide a range of problems to test the capabilities of the orderings. As will be shown in the following sections, they suffice to show that the "general purpose" sparse matrix ordering heuristics do differ in their robustness. Our goal of providing relative evaluation of the orderings can be met with these problems; they provide only a direction towards our goal of providing means to anticipate the expert's analysis of problems.

1.4. Limitations. In many ways any study of this sort must be incomplete and limited, and this study is no exception. The size of the test set is inadequate to draw deep conclusions. For example, we have only two sources of chemical engineering problems, and a very small number of problems in other significant disciplines. This is particularly unfortunate in limiting the value of statistical analyses of certain density properties of the matrices. These analyses indicate differences among the problem classes that correspond to differences in ordering behavior. But they cannot be deemed to be significant; instead, they are only interesting and provocative. They are discussed in more detail in the following sections.

Our selection of algorithms is also limited. We have omitted the Lin and Mah algorithm because of the difficulty in incorporating it. There may be other algorithms in other disciplines, which are effective, but unknown to the authors. Several reasons led us to omit a post-processing phase in the factorization to stabilize numerically the P^4 ordering. One was our desire to use a common factorization module for all orderings to ensure that the results were comparable. Our initial testing indicated that the P^4 ordering was less effective in general than Markowitz; since the required pivoting can be expected to increase the fill we felt the results would not change any conclusions. This is reinforced by the results from [15], [26], [27], which indicate that the fill in P^4 orderings is very sensitive to the number of column interchanges. However, these results are not general; the arguments that fill will increase are substantial, but not proofs. As the results stand, our P^4 results are at least parallel to those from P^5 , which we do not yet know how to stabilize efficiently.

A final limitation on this work is its emphasis on generality. As such it represents a bias against the importance of nearly structurally symmetric problems. Interesting problems not addressed in this study because of this bias are discussed in § 4.

2. Symbolic properties. Among the purposes of this study were the investigations of several purely symbolic aspects of general unsymmetric sparse matrix reorderings. These included

—Determining if any or all of the ordering algorithms are generally successful in obtaining low fill and low work factorizations;

—Determining if there are simple statistical measures of matrix properties that can predict how well specific matrices will be reordered.

Each of these leads to a series of further questions to be answered, which are addressed in this section. The discussion of storage and work reduction is addressed first, followed by the suggestive work on predicting which problems will be difficult.

2.1. Symbolic (preordering) effectiveness of ordering heuristics. A sparse factorization, one with low fill and low operation counts, is a paramount goal of these reordering heuristics. Several factors in addition to the heuristics contribute to achieving this goal and to our being able to measure the results. One of these is the effectiveness of the block triangularization preordering. Failures of the evaluation code are another. The evaluation program was not designed to extend a symbolic factorization beyond a structurally zero diagonal element, if such occurred with the P^4 ordering. Further, the MA28 underpinnings of the evaluation code are not capable of adjusting space among the three required and separate workspaces when the space in just one of the workspaces was exhausted. Several of the largest (and most expensive) evaluation runs failed when one of the MA28 workspaces was inadequate. Unfortunately, our resources were also inadequate to allow repeated runs to obtain a proper distribution of space. For these reasons, analyses of all of the matrices were not possible.

Block triangularization. All of the ordering heuristics in this study were preceded by a preordering phase in which a permutation to the finest block lower triangular form of the matrix was found. An implementation of Tarjan's algorithm in the World Bank P^4 code was used for this reduction in conjunction with the P^4 and P^5 heuristics. The separate option in MA28, also Tarjan's algorithm, was used when the Markowitz heuristic was applied. The uniqueness of the reduction up to permutations within the blocks (see [2]) guarantees that the results of the two different implementations are comparable. In both cases, contiguous diagonal blocks of order one were coalesced into larger, triangular, blocks.

Tables 2.1 and 2.2 demonstrate the effect of block triangularization on our test matrices. The first table shows the average number of equations, average number of

TABLE 2.1
Block triangularization statistics for the classes of test matrices.

Problem class	No. of matrices	Average order	Ave. no. diagonal blocks	Ave. no. nontrivial diag. blks.	Ave. percent of columns in nontrivial diag. blks.	Ave. size of nontrivial blocks
Chemical engineering	16	506.4	6.9	3.3	62.1	122.5
Linear programming	16	677.4	21.8	10.50	29.4	19.0
Simulations	11	441.6	1.4	1.00	99.9	441.3
PDE grids	7	959.0	1.7	1.29	93.4	694.8
Miscellaneous	6	97.5	2.2	1.33	96.3	69.4
Optimal power flow	2	4929.0	2.5	1.00	92.6	4565.5
Overall	58	706.1	8.7	4.33	68.6	235.8

TABLE 2.2
Significance of block triangularization for the classes of test matrices.

Problem class	Matrices with no reducibility	Matrices with insignificant reducibility	Matrices with significant reducibility
Chemical engineering	0	2	14
Linear programming	0	0	16*
Simulations	7	4	0
PDE grids	5	0	2
Miscellaneous	3	2	1
Optimal power flow	0	2	0
Overall	15	10	33

* Includes 5 lower triangular matrices.

diagonal and nontriangular diagonal blocks, the average size of the nontriangular blocks, and the average percentage of columns in nontriangular diagonal blocks. These numbers are presented by problem class as well as the overall numbers, which highlights a clear, and understandable difference between the classes of problems. The partial differential equation matrices arise from grid discretizations, which are irreducible unless there are inactive cells or internal boundary conditions. The simulation and optimal power flow problems are representative of models of closed loop systems, but not of other classes of simulations. The problems from the other disciplines show large reductions in the order of the matrices that actually must be factored. Nothing in our results argues against the conventional wisdom that this reordering should be performed, unless there are reasons to believe that the problem is irreducible. Summary figures, by origin, are given in Table 2.2, which shows the distribution of matrices, for which the block triangularization significantly reduced the effective order of the problem.

Fill characteristics. Statistics on the factors of the matrices were obtained from the common symbolic factorization routine adapted from MA28A. The statistic denoted by "explicit fill" is the increase in the number of nonzeros stored to represent the factors L and U over the number of nonzeros in the original matrix A . Since all of the matrices were partitioned into a block lower triangular form, the explicit fill is exactly the collective fill in the nontriangular diagonal blocks.

Within the block lower triangular form each diagonal block acquires a structure from the ordering heuristic. The Markowitz ordering is generally unstructured, with

some increases in density towards the lower right corner of the factors. The P^4 heuristic confines the fill to the spike columns, which can be used to simplify the factorization or auxiliary computations in some contexts. Use of this may change the data structures, but does not change the fill statistics. The diagonal blocks resulting from the P^5 ordering have a very explicit structure; they are block bordered lower triangular matrices. This structure allows us to use implicit block factorization techniques, in which the only fill occurs in the final diagonal subblock of each (outer) diagonal block. This is discussed in detail in [14]. The actual factorization implemented in our evaluation program used the standard explicit approach, which gave us explicit fill statistics for this algorithm. The special character of the implicit factorization and statistics extracted on the partitioning obtained by the P^5 code allowed us to compute the fill that would have been required by an implicit factorization. We refer to this statistic as "implicit fill." Unfortunately, storage requirements and operation counts for this alternate factorization were not available.

The Markowitz ordering was used in two modes, one as pure a Markowitz ordering as can be obtained from MA28, the other as a hybrid ordering combining the Markowitz heuristic with limited pivoting for numerical stability. Although the discussion of numerical properties is the subject of the next section, the fill statistics for the stabilized version of the Markowitz heuristic are presented here for comparison.

Table 2.3 gives the average of actual fill of the various orderings with respect to the various problem classes. Table 2.4 gives the average percentage of fill relative to the original number of nonzero entries for the same problems.

TABLE 2.3
Average of actual fill for the classes of test matrices.

Problem class	No. of matrices	MA28 (.0)	Explicit fill			Implicit fill
			MA28 (.1)	P^4	P^5	P^5
Chemical engineering	10	126.2	142.4	244.1	262.7	63.5
Linear programming	6	17.0	17.7	23.2	23.8	3.2
Simulations	4	2227.5	2276.5	2099.0	3140.8	485.3
PDE grids	4	12866.5	9935.0	10356.0	10818.5	1440.5
Miscellaneous	6	170.0	186.2	328.0	416.0	53.3
Optimal power flow	2	18518.0	18561.0	—	218466.0	29495.5
Overall	30	2092.0	1716.4	1812.3	2036.8	289.2

TABLE 2.4
Average percentage of relative fill for the classes of test matrices.

Problem class	No. of matrices	MA28 (.0)	Explicit fill			Implicit fill
			MA28 (.1)	P^4	P^5	P^5
Chemical engineering	10	22.1	24.5	35.9	41.5	11.4
Linear programming	6	.5	.5	.6	.6	.1
Simulations	4	265.6	268.3	259.7	380.6	59.3
PDE grids	4	182.5	157.0	176.0	184.9	23.7
Miscellaneous	6	37.3	41.0	68.0	88.0	14.8
Optimal power flow	2	55.9	56.0	—	659.1	89.0
Overall	30	74.6	73.2	83.8	106.9	17.9

These statistics were restricted to the subset of problems where both the ordering and the symbolic factorization were computed. Five of the test problems were excluded because the storage demands for the Markowitz code were excessive, six failed for the same reason for the P^4 heuristic, and the same failure occurred seven times for the P^5 code. The most frequent case of failure in the symbolic factorization was due to the P^4 ordering, which produced structurally zero diagonal entries for twenty-two of the problems. This is itself a surprising result. Table 2.5 contains the distribution of these P^4 failures by problem origin. Overall these failures reduced the set of test matrices for which comparable results were obtained from the total of fifty-eight to thirty matrices.

TABLE 2.5
Distribution of P^4 failures for the classes of test matrices.

Problem class	No. of storage aborts	No. of symbolic factorization aborts	No. of numeric aborts	No. of numerically unstable factors*	No. of complete successes
Chemical engineering	0	6	4	1	5
Linear programming	0	10	0	0	6
Simulations	5	2	0	2	2
PDE grids	1	2	3	1	0
Miscellaneous	0	0	0	2	4
Optimal power flow	0	2	0	0	0
Overall	6	22	7	6	17

* As defined in § 3 as failures.

Because of their unusual fill statistics the results for the optimal power flow problems are listed separately in Tables 2.4 and 2.5. Since P^4 failed for these problems, the results for these problems are not included in the overall figures.

The following general observations can be made on the data in the above tables on fill statistics. They are

—The cost in increased fill of incorporating numerical safeguards in the Markowitz heuristic is slight. The fill for MA28 with a nonzero pivot tolerance is never much more than the fill for the pure Markowitz heuristic; on average it is actually less.

—The explicit fill for P^4 , even for the problem classes where P^4 is commonly used, is almost always greater than Markowitz with pivoting.

—The explicit fill for P^5 is always worst.

—The implicit fill for P^5 is always less, sometimes significantly so, than the explicit fill of all the other orderings, except for the optimal power flow problems.

—The problem classes with the largest amount of fill are the PDE problems, the simulation problems, and the optimal power flow problems. These are also the classes for which the block triangularization was least effective.

Overall, both the Markowitz and the implicit P^5 heuristics perform well across a spectrum of problems. The P^4 heuristic is not much worse in its performance when it works, but it frequently fails as a strict preordering heuristic because of structural zero pivots. The implicit fill results indicate that further research on the P^5 algorithm may prove fruitful, although the large amount of implicit fill for the optimal power flow problems indicates that P^5 may be far less a general purpose reordering than originally

expected. However, as is discussed in § 3, a numerically robust version of the P^5 algorithm does not yet exist. By contrast, the Markowitz heuristic, constrained for stability, functions well in general.

The last observation is that the PDE, simulation, and optimal power flow problems are the hardest of these test problems. It was expected that the PDE problems would have large fill as they arise from well connected grid discretizations, which yield irreducible matrices. Similarly the classes of simulation problems and of optimal power flow problems show insignificant irreducibility (see Table 2.2) and the comparatively large amount of fill came as no surprise. What was surprising was the fact that both problem classes also exhibited poor stability (see Table 3.2). That these two factors were correlated was unexpected. Although this is too small a sample to draw conclusions, this correlation should be investigated further.

There are other important statistics on the performance of the ordering heuristics, in particular, factorization storage requirements and operation counts. These statistics were not available for the implicit factorization for the P^5 heuristic. They are correlated with the fill statistics for the explicit factorization, since we used the same data structures and factorization routines throughout. For these reasons, we confined our attention to performance in terms of fill.

2.2. Statistical predictors for fill. Sparse matrix ordering heuristics are not universally effective. The relative performance between heuristics usually depends on the type of problem being reordered. Certain heuristics are known to be best for certain problem classes, and not best for others. The algorithms and test cases in this study show this behavior only to a limited degree; the more extensive set of orderings for symmetric matrices illustrates this point more. One of our goals, however, was to investigate the extent to which problem origin affected the outcome of our unsymmetric ordering heuristics and to search for predictors for the fill that occurs. Although our ordering heuristics fail to show spectacular relative differences, our classes of test matrices do. The PDE matrices and the simulation matrices are clearly more difficult to reorder well than are the other classes. Can this be predicted from observable properties of the original matrix?

We measured a number of variables that may indicate properties of the test matrices. For simple density considerations we computed and saved the number of nonzeros in the original matrix, and also the number in the lower and upper triangles of the original matrix, respectively. We attempted to measure the regularity of the distribution of nonzeros by computing the minimum, the maximum and the standard deviation of the nonzeros by row and by column respectively. Several measures of matrix structural symmetry were developed that indicate how close a matrix is to structural symmetry. The measure used in the analyses in the section is the percentage of entries in the matrix for which the corresponding entry in the transpose of the matrix is also nonzero. Statistics were kept on the number, size and average size of nontrivial diagonal blocks found by the block triangularization.

These measures of density, regularity and symmetry were compared by class with the performance of the ordering heuristics. The two classes with the highest relative fill contrasted with the other classes, according to these measures. The class of PDE matrices, for example, has the highest average symmetry ratings, has high initial density, and is more regular than the other classes. The simulation matrices on the other end are much less regular. Their density, as measured by the average of nonzeros by row (column), is typical for the test set, but the nonzeros are arranged irregularly. They have high maximum column counts and a high standard deviation of column counts.

By contrast, the linear programming matrices have high maximum and standard deviation of row counts, with similar density.

Table 2.6 contains the measures for density, regularity and symmetry. On the basis of these tables, the PDE problems stand out for the reasons suggested above. The LP and simulation problems stand out for their irregularity, and there are few other obvious differences. The major difference between the LP and the simulation problems would appear to be the degree to which they split into smaller irreducible problems. We might have left this topic with the conclusion that this more costly measure represents the most significant difference between the problem classes. Instead, we proceeded with further statistical analyses.

TABLE 2.6
Average density, symmetry and regularity measures for the classes of test matrices.

Problem class	Average nonzeros per row/column	Nonzeros per row maximum	Nonzeros per column standard deviation	Nonzeros per column maximum	Nonzeros per column standard deviation	Overall symmetry rating
Chemical engineering	3.84	11.81	2.41	22.88	3.51	0.05
Linear programming	5.49	256.38	11.83	20.50	4.64	0.01
Simulations	5.68	7.73	1.28	200.36	8.94	0.46
PDE grids	10.14	13.71	1.96	13.71	1.89	0.99
Miscellaneous	5.57	30.17	4.30	32.33	5.17	0.54
Optimal power flow	6.70	35.50	3.00	28.00	3.40	0.20
Overall	5.69	81.45	4.95	55.93	5.57	0.27

We performed a series of regression analyses to determine how well the measures discussed above predict fill and relative fill. We used the widely available SPSS statistical package [25] to compute step-wise regressions of the fill statistics for each ordering against the symmetry rating, the standard deviations of column and of row counts (measures of regularity), the number of diagonal blocks produced in the block triangularization, and the original number of nonzeros. The relative fill statistics were regressed against the order of the matrix, the symmetry, regularity and reducibility measures, and the average number of original entries per row (column).

Table 2.7 contains a summary of these analyses. Only the variables that were significant in the regression equations are listed. It comes as no surprise that the original number of nonzeros is significant in estimating the (additional) fill. It is surprising that the symmetry ratings and the regularity measures are often significant, particularly in predicting relative fill. The results of these regressions indicate that the ordering heuristics are more effective for "general" problems, which are far from symmetric and which are irregular, than they are on problems with more structure. This conclusion is not new, but the fact that these structural measures have predictive power for the conclusion is.

It would be inappropriate, if not foolhardy, to claim statistical significance for these results. The usual normality assumptions would be rather difficult to justify for our small test collection. In addition, a linear model, as is assumed by these regressions, is unlikely. Only very special sparse matrices are known to have only a linear dependence on such parameters as order. Most matrices show a faster, nonlinear, growth. On the other hand, these results do suggest that further analysis may confirm what is now anecdotal: that general orderings for unsymmetric matrices perform poorly on symmetric matrices and matrices with a regular structure.

TABLE 2.7
Regressions for fill and relative fill.

	Variable	Fill Coefficient	Significance	Variable	Relative fill Coefficient	Significance
Markowitz	Nonzeros	3.2	.00	Symmetry	.016	.01
	Symmetry	48.4	.01	S.d. rows	-.104	.01
	S.d. rows	-32.0	.01			
	No. of blocks	-129.0	.01			
Goodness of fit		0.91		0.46		
Markowitz (with stability tolerance 0.1)	Nonzeros	2.5	.00	Symmetry	.015	.01
	S.d. rows	-311.2	.02	S.d. rows	-.095	.01
	Symmetry	55.2	.00			
	No. of blocks	-115.8	.01			
Goodness of fit		0.89		0.43		
P ⁴	Nonzeros	2.6	.00	S.d. rows	-.112	.04
	Order	-9.2	.00			
	S.d. columns	-260.0	.00			
	No. of blocks	-51.1	.05			
Goodness of fit		0.98		0.46		
P ⁵ (explicit factorization)	Order	88.0	.00	Order	.005	.00
	S.d. rows	-2713.1	.03	S.d. rows	-.254	.00
				Symmetry	.019	.02
Goodness of fit		0.83		0.78		
P ⁵ (implicit factorization)	Nonzeros	0.9	.00	Order	.003	.00
				S.d. rows	-.022	.04
				S.d. columns	-.016	.04
Goodness of fit		0.71		0.35		

Variables included in regression: order, symmetry rating, standard deviation of column counts, standard deviation of row counts, number of diagonal blocks in block triangular form, and either original number of nonzeros (for fill) or average number of nonzeros per row/column (for relative fill). Only variables which are significant at a 0.05 level are listed. Other variables were not significant at this level. Variables are listed in the order that they entered the stepwise regression.

Goodness of fit statistic is R square (norm of residual for fit/norm of residual for constant model).

3. Numerical properties. The basic ordering heuristics, Markowitz, P^4 and P^5 , were all originally proposed as *preorderings*, that is, methods for reordering sparse matrices based only on their nonzero structures, but not on their numerical values. Unsymmetric sparse matrices cannot, in general, be factored accurately and stably without some pivoting based on the numerical entries. This study included an investigation of several questions that address the value of these heuristics as preorderings, and as bases for stable algorithms. Some of these questions were

—Do the problems differ in their numerical behavior because of their differing origins? Are there classes of problems for which preordering the matrices is sufficient?

—What are the costs of providing modified ordering heuristics that are stable?

—How reliable are a posteriori error bounds on the error in the factorization?

3.1. Numerical stability without pivoting. Three different outcomes may result from the numerical factorization of a preordered sparse matrix: a sufficiently accurate factorization may be found, a factorization that is inaccurate may be computed, or the factorization may break down because an exactly zero pivot may be encountered. The last case is usually thought of as unlikely in numerical analysis, but it is, in fact, encountered frequently in this particular area. We shall first investigate how frequently the factorization simply cannot be computed.

Exact zero pivots result from the interaction of small numbers of equations with closely related entries from approximate models. For example, one test matrix caused the P^5 factorization to fail because the heuristic had produced a two by two diagonal block, all of whose entries were the square root of two to machine precision. The numerical factorization of this matrix must conclude that this submatrix has rank one. The Markowitz strategy and the P^4 heuristic can fail in the same way. The P^4 heuristic, as discussed earlier, can also fail by producing a structurally zero diagonal element.

The comparison of the preordering stability properties is complicated by the fact that the MA28 implementation of the Markowitz strategy cannot break down in this fashion. Since the code computes the numerical factorization during the ordering process, it can detect an exactly zero diagonal element. When this occurs, the Markowitz ordering is perturbed; the pivot chosen is some truly nonzero entry with the best Markowitz count obtainable in these circumstances. In this sense, MA28 does not provide a pure Markowitz ordering. By contrast, the numerical factorization for P^4 and P^5 will break down if an exactly zero pivot is encountered.

Table 3.1 provides a partial answer to the value of these heuristics as preorderings. The number of successful symbolic and numerical factorizations obtained for each ordering are listed. The MA28 code cannot fail to provide some numerical factorization

TABLE 3.1
Success of symbolic and numerical factorization for the classes of test matrices.

Problem class	No. of matrices	MA28	P^4		P^5	
			Symbolic	Numeric	Symbolic	Numeric
Chemical engineering	16	16	10	6	15	8
Linear programming*	16	16	6	6	16	12
Simulations	11	9	4	4	6	6
PDE grids	7	4	4	1	6	1
Miscellaneous	6	6	6	6	6	6
Opt. power flow	2	2	0	0	2	0
Overall	58	53	30	23	51	33

* These numbers include 5 lower triangular matrices on which all algorithms were successful.

if it has space for the symbolic factorization, so only a single column is listed for this heuristic. These operations are listed separately for the P^4 and P^5 heuristics. As discussed in § 2, the symbolic factorization for P^4 can fail for two reasons, for lack of space or because of structurally zero diagonal elements. The difference between the failures for the symbolic and numerical factorizations of these two algorithms demonstrate how frequently exact zero diagonal elements are encountered. All told, the P^4 heuristic produced twenty-two structurally zero diagonal entries and seven "accidental," but exact, zero entries; the P^5 heuristic produced such accidental zero entries eighteen times.

It is certainly naive to assume that nonbreakdown of the numerical factorization is, in fact, success. The results in Table 3.1 simply indicate that these preorderings are not robust in the weakest sense. Table 3.1 does not measure numerical stability. For example, the norm of the error, $\|E\| = \|A - LU\|$, in the LU decomposition relative to A was of the order of 10^{30} for the only PDE problem "successfully" factored with the P^4 ordering. In order to obtain a better picture on the quality of performance of the algorithms, we used this relative measure of the error to rate the stability as follows:

$\ E\ /\ A\ \leq 10^{-8}$	good stability,
$10^{-8} < \ E\ /\ A\ \leq 10^{-4}$	fair stability,
$10^{-4} < \ E\ /\ A\ \leq 1$	poor stability,
$\ E\ /\ A\ > 1$	failure.

Tables 3.2 through 3.4 summarize the stability results, using the qualitative descriptors above. The three tables provide a comparison across algorithms.

The numbers in these tables allow several conclusions about the relative performance of the algorithms with regards to stability. The tables suggest that the Markowitz

TABLE 3.2
Stability rating of MA28 with 0.0 pivoting threshold for the classes of test matrices.

Problem class	No. of matrices	No factors	Fail	Poor	Fair	Good
Chemical engineering	16	0	0	2	1	13
Linear programming	16	0	1	2	1	12
Simulations	11	2	8	0	1	0
PDE grids	7	3	3	0	0	1
Miscellaneous	6	0	0	1	1	4
Opt. power flow	2	1	1	0	0	0
Overall	58	6	13	5	4	30

TABLE 3.3
Stability rating of P^4 algorithm for the classes of test matrices.

Problem class	No. of matrices	No factors	Fail	Poor	Fair	Good
Chemical engineering	16	10	1	0	0	5
Linear programming	16	10	0	0	0	6
Simulations	11	7	2	0	1	1
PDE grids	7	6	1	0	0	0
Miscellaneous	6	0	1	1	0	4
Opt. power flow	2	2	0	0	0	0
Overall	58	35	5	1	1	16

TABLE 3.4
Stability rating of P^5 algorithm for the classes of test matrices.

Problem class	No. of matrices	No factors	Fail	Poor	Fair	Good
Chemical engineering	16	8	0	0	1	7
Linear programming	16	4	1	2	1	8
Simulations	11	5	2	0	3	1
PDE grids	7	6	1	0	0	0
Miscellaneous	6	0	2	0	1	3
Opt. power flow	2	2	0	0	0	0
Overall	58	25	6	2	6	19

ordering yields generally better results than either P^4 or P^5 . In thirty-four cases the stability rating of MA28 was either good or fair, whereas the corresponding numbers are only 17 for P^4 and 25 for P^5 . This is somewhat misleading, since the modification discussed above allows the MA28 Markowitz code to provide some factorization for almost all of the test problems. In fact, on comparable problems, the Markowitz ordering does often fare better, but this is not true for the simulation matrices. It is clearer that the P^5 ordering is an improvement over the P^4 ordering in terms of numerical stability. With the exception of one test matrix, the stability rating of P^5 is at least as good as that of P^4 . For all of the orderings there are many matrices for which preordering alone is not satisfactory.

There are clear differences in numerical stability between the problem classes. The high rates of failure in the PDE problems, in the simulation problems and in the optimal power flow problems of all three algorithms confirm the observation made in the last section that these problems apparently are more difficult than the other matrices in the study. All three algorithms almost always failed for the PDE problems. The simulation problems are the only problems where P^4 and P^5 perform better than MA28, yet all three algorithms have a high rate of failure. The linear programming problems are not as difficult as the other classes, and yet the P^4 and P^5 heuristics succeed on only six and eight of these problems, respectively. Of these, five are triangular, for which the decomposition is trivially stable. This result is somewhat surprising, since the P^4 heuristic was designed for LP problems. On none of the classes do the P^4 and P^5 algorithms provide satisfactory results more than half the time. Hence there appears to be no particular problem class for which either P^4 or P^5 orderings are more appropriate than the Markowitz ordering for numerical stability. It is also clear from the overall failure rate of all three heuristics that any general use of these orderings requires modification for stability.

3.2. Costs and effects of stability modifications. The heuristic orderings must be modified to provide numerical stability in the factorizations. The cost of incorporating such modifications, and its effects on stability and fill are the topic of this section.

The discussion of the Markowitz ordering in § 1 concluded that, with little additional cost, the ordering heuristic could have knowledge of the numerical entries in each current reduced matrix. The MA28 code demonstrates that incorporating some control on element growth is possible with essentially the data structures already required for the Markowitz ordering. Duff's discussion of the development of the code [3], [4] does indicate that the development of a practical code was not trivial. However,

from the perspective of the user, the changes needed to compute a stable factorization with this code are trivial.

The Harwell MA28 Markowitz code controls the growth of elements in the factorization, and hence instability, by using a growth or pivoting threshold that can take values between zero and one. A growth tolerance of one corresponds to a pivot selection based essentially on numerical considerations; it is in fact Gaussian elimination with partial pivoting, with pivots for sparsity preservation chosen only from the largest elements in rows. The experience of the Harwell designers and others has been that this is too restrictive to provide practical sparse factorizations, and that a weaker control still suffices to control instability. We chose, with some assistance from the standard recommendations for the code, to use a growth threshold of one tenth (0.1). This results in the rejection of a Markowitz pivot only if there is an element in the same row at least ten times larger than the proposed pivot. It also potentially allows an order of magnitude growth in some of the elements of the factorization at each step.

Table 3.5 provides the basis for comparing the Markowitz ordering with limited stability control to the preorderings discussed in the previous section. Of the fifty-three matrices for which numerical factorizations were computed, in only five did the MA28 code fail to obtain a "good" factorization. It is a surprising fact that the rather substantial growth allowed at each step does not accumulate. It is also surprising, although it is consistent with previous published discussion of this algorithm, that the cost of obtaining a stable factorization is little more than the cost of the Markowitz preordered factorization. Tables 2.3 and 2.4 illustrate the small (if any) increase in fill associated with the restrictions placed on the ordering. There is a corresponding increase in the number of operations required to factor the matrix, and also in the time for ordering the matrix. The latter is beyond the scope of this study; some examples are given in [3], [4].

TABLE 3.5
Stability rating of MA28 with 0.1 pivoting threshold for the classes of test matrices.

Problem class	No. of matrices	No factors	Fail	Poor	Fair	Good
Chemical engineering	16	0	0	0	0	16
Linear programming	16	0	0	0	0	16
Simulations	11	2	0	0	0	9
PDE grids	7	3	0	0	0	4
Miscellaneous	6	0	0	0	0	6
Opt. power flow	2	0	0	0	0	2
Overall	58	5	0	0	0	53

Modifications to stabilize the ordering provided by the P^4 heuristic were discussed in the introduction. Growth and instability are controlled by a very similar threshold for required pivoting. This frequently used technique requires no modification to the heuristic itself, but does require increased complexity of the factorization data structures. In particular, the space requirements for the factorization will not be known until the factorization is completed. Although this increases the overall complexity of the process, it is, in fact, still less complex than the Markowitz ordering with control.

As was discussed earlier, limitations on this project led to the omission of a stabilized P^4 factorization from the evaluation code. The only results known to us that compare the increased cost in fill for stabilizing the P^4 heuristic are contained in [15],

[26], [27]. These results indicated that the amount of increased fill was more sensitive to the growth threshold than is the case for the Markowitz code. In particular, Saunders indicated that a threshold of 0.1 was substantially too large, that more growth would have to be allowed if sufficiently sparse factors were to be obtained. This could well result in numerically less satisfactory factorizations. There are arguments that explain why the modification to P^4 could result in rapid increases in fill. Since the fill for the stabilized Markowitz code with threshold of one tenth is often less than the fill for the unstabilized P^4 code there appear to be few reasons to study a stabilized P^4 code for great reduction of fill.

In contrast, the implicit fill statistics for the P^5 ordering are very small, except for the optimal power flow problems. They suggest that a stable version of this ordering and factorization would be quite valuable for certain problem classes if they have similar fill statistics. Unfortunately, we do not know how to obtain such a factorization scheme. We shall discuss why we believe this to be a difficult problem.

Let us note at the outset that it is not hard to find a stable factorization for the P^5 heuristic: the ideas used in conjunction with P^4 work also in this context. Note however that the P^5 ordering induces a special structure, a block bordered triangular form, to any irreducible matrix. Specifically, an irreducible matrix A , of order n , will be partitioned as follows:

$$(1) \quad A = \begin{bmatrix} D & B \\ C & S \end{bmatrix},$$

where the $(n-p)$ by $(n-p)$ matrix D is lower block triangular:

$$(2) \quad \begin{bmatrix} D_{11} & & & & \\ D_{21} & D_{22} & & & 0 \\ \vdots & \vdots & \ddots & & \\ D_{n1} & D_{n2} & \cdots & \cdots & D_{nn} \end{bmatrix}.$$

We assume that the number of rows/columns p in the border blocks is small as compared to n , and that both B and C as well as the offdiagonal blocks in D are sparse.

It is possible to incorporate pivoting within the matrices D_{ii} and the matrix S without destroying the block partitioning. Such pivoting may be used to improve the stability of the factorization. It is, however, not enough to guarantee stability. Generally, stability requires that columns be interchanged between D and B (or that rows be interchanged between D and C). The ideas used for stabilizing the P^4 ordering interchange columns.

The crux of the matter is that the interchange of columns of D and columns of B may well destroy the block triangular form of D . If it does, we do not know how to perform the factorization implicitly. If we must form the factorization explicitly, the fill will be too great, even without the increases we expect from perturbing the ordering. The difference between the fill of implicit and explicit factorizations is evident from Table 2.3. We shall illustrate the problem of instability with a model problem.

Consider the following matrix of order n :

$$\begin{bmatrix} -4 & 1 & 0 & \cdot & \cdot & 0 \\ 1 & -4 & 1 & \cdot & \cdot & \cdot \\ 0 & 1 & -4 & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & 1 & -4 & 1 \\ 0 & \cdot & \cdot & 0 & 1 & -4 \end{bmatrix}.$$

That this matrix is symmetric is irrelevant to this discussion, as is the fact that it can be factored stably without pivoting as it is written. The matrix, after it has been reordered with the P^5 heuristic, is given below:

$$\begin{bmatrix} 1 & 0 & \cdot & \cdot & \cdot & -4 \\ -4 & 1 & 0 & \cdot & \cdot & 1 \\ 1 & -4 & 1 & 0 & \cdot & 0 \\ 0 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & -4 & 1 & 0 \\ 0 & \cdot & 0 & 1 & -4 & 0 \end{bmatrix}.$$

Note that the border has only a single column, so we will denote it as a vector b . The leading principal block D is unit lower triangular. The implicit fill is exactly the order of the block S , one, independent of the order, n , of the original matrix.

Gaussian elimination on the reordered matrix requires, either explicitly or implicitly, the solution of the triangular linear system $Dy = b$. It is easy to show that $y_1 = -4$, $y_2 = -15$, $y_3 = -56$ and that the elements of y thereafter grow by a factor of about $2 + \sqrt{3}$ at each step. This is unacceptable growth, which leads to serious stability problems. Element growth similar to this was encountered in the factorizations of the PDE matrices in our test collection.

The model problem explains the nature of the serious stability problems associated with the P^5 ordering. We have investigated other approaches to avoid element growth while preserving the block bordered triangular form. Approaches based on the Sherman-Morrison-Woodbury updating form are possible, but they appear to be equivalent to the original in terms of growth, that is, to be only implicit reformulations that must solve a similar linear system at some point. Another attempt to circumvent the element growth in the formation of the Schur complement is to use an orthogonal factorization of the matrix instead of using block Gaussian elimination. Since the matrix is almost in lower triangular form, one is tempted to believe that a strategy based on the LQ factorization might be successful. This is unlikely, for the following reason. The matrix L in the LQ factorization is the Cholesky factor of the normal equations, i.e. the Cholesky factor of

$$\begin{bmatrix} DD^T + BB^T & DC^T + BS^T \\ CD^T + SB^T & CC^T + SS^T \end{bmatrix}.$$

There is no reason to suspect that the partitioning obtained by P^5 is a good reordering for this symmetric matrix. Indeed, there are reasons to suspect the opposite: the heuristic tends to put denser columns into the border, which will cause the leading principal block in the above to be substantially more full.

It should be mentioned that Heath [16] suggests a reordering scheme for the solution of particular sparse linear least squares problems by an orthogonal factorization, which is closely related to some ideas expressed here. When full rows are present, Heath proposes to reorder them last and use block Gaussian elimination to solve the complete problem. Even though the leading sparse part of the matrix is factored with stable orthogonal techniques, the use of block Gaussian elimination may lead to the same stability difficulties encountered with the P^5 algorithm.

3.3. Evaluation of measures of stability. Numerical stability is a concern, even with a factorization that provides some control over element growth. A side issue we addressed was the reliability of various estimates of the error in the actually computed factorization.

The error in the factorization is defined by $E = A - LU$. The scalar that best summarizes this error is the relative error norm, $\|E\|/\|A\|$. Since computing E , or $\|E\|$, is often unacceptably expensive, several heuristic algorithms have been suggested. These estimates, which have dramatically different costs, are

(1) Error in the solution $\|x_{\text{true}} - x\|_1$ where $x_{\text{true}} = (1, 1, \dots, 1)^T$, $b = Ax_{\text{true}}$, and x is the computed solution;

(2) Erisman-Reid error bound [13]

$$\max_{i,j} |e_{ij}| \leq 3.01 \epsilon m (\max_{i,j} |a_{ij}| + \|L\|_1 \|U\|_1)$$

where ϵ = machine epsilon, and m is the maximum number of operations needed to compute one entry of L or U ;

(3) Wilkinson error bound [28]

$$\max_{i,j} |e_{ij}| \leq 3.01 \epsilon m (\max_{i,j,k} |a_{ij}^{(k)}|).$$

The values obtained for these estimates and made relative to $\|A\|$ are summarized in Tables 3.6–3.8. These tables give the average values of the base 10 logarithms of the estimates for all of the obtained numerical factorizations. The average of the logarithms of the actual relative error norm are given for comparison, in Table 3.9.

The quantity (3) is a bound on the actual error; the quantity (2) is a bound on (3). We expect, and find, that the Erisman-Reid bound is larger, more pessimistic about the success of the factorization, than the Wilkinson bound. This is, in turn, larger than the actual error. On average, the Wilkinson bound is larger than the actual error

TABLE 3.6
Solution error for the classes of test matrices.

Problem class	MA28 (.0)	MA28 (.1)	P ⁴	P ⁵
Chemical engineering	-10.1	-13.9	-10.7	-10.4
Linear programming	-8.0	-13.1	-14.1	-9.5
Simulations	+19.3	-11.6	-0.2	+1.1
PDE grids	+11.6	-19.0	+18.3	+22.2
Miscellaneous	-8.9	-13.1	-6.0	-4.5
Opt. power flow	+14.8	-11.6	—	—
Overall	-2.1	-13.5	-7.2	-5.9

TABLE 3.7
Erisman-Reid error bound for the classes of test matrices.

Problem class	MA28 (.0)	MA28 (.1)	P ⁴	P ⁵
Chemical engineering	-3.4	-10.4	-6.3	-5.6
Linear programming	-3.6	-11.2	-11.2	-4.5
Simulations	+23.9	-8.9	+7.8	+8.4
PDE grids	+13.6	-8.9	+22.2	+22.2
Miscellaneous	-2.1	-10.4	-0.3	+4.4
Opt. power flow	+28.1	-7.8	—	—
Overall	+3.3	-10.2	-2.3	+0.0

TABLE 3.8
Wilkinson error bound for the classes of test matrices.

Problem class	MA28 (.0)	MA28 (.1)	P ⁴	P ⁵
Chemical engineering	-8.7	-12.4	-10.0	-9.9
Linear programming	-7.1	-12.7	-11.5	-7.5
Simulations	+18.2	-10.6	+0.8	+2.5
PDE grids	+13.6	-11.8	+22.2	+22.2
Miscellaneous	-7.5	-11.9	-4.8	-3.2
Opt. power flow	+16.6	-10.8	—	—
Overall	-1.2	-12.0	-3.7	-4.6

TABLE 3.9
Actual error for the classes of test matrices.

Problem class	MA28 (.0)	MA28 (.1)	P ⁴	P ⁵
Chemical engineering	-11.5	-15.3	-12.7	-13.0
Linear programming	-10.3	-14.4	-14.6	-10.1
Simulations	+17.1	-13.9	-0.8	-0.7
PDE grids	+13.3	-13.8	+22.2	+22.2
Miscellaneous	-9.4	-14.5	-6.7	-5.2
Opt. power flow	+13.8	-13.7	—	—
Overall	-3.5	-14.5	-8.0	-7.2

by three orders of magnitude. This is a large error in an absolute sense, but it is unlikely to cause a satisfactory factorization to be rejected. On the other hand, the Erisman-Reid bound is large enough so that some factorizations might be unduly rejected. The results for this bound on the LP bases as ordered by the pure Markowitz ordering are an example. However, none of the factorizations with the stabilized Markowitz ordering would be rejected. The Erisman-Reid bound appears to be a reasonable tool in conjunction with this particular code. (Note that the version of this bound for symmetric matrices is usually considerably more sharp.)

The error in the solution of a specific equation, (1), is only an estimate, not a bound on the factorization error. While it is usually a better estimate of the error than either of the bounds, it does have the potential to mislead. The PDE matrices ordered by the stable Markowitz ordering are a particular case where this estimate indicates that the factorizations are substantially more accurate than they really are.

The four potential measures or estimates of factorization error differ greatly in cost. The cost of computing the error in a specific equation is only a matrix multiply to generate a right-hand side and a matrix solve. The Erisman-Reid error bound requires a single scan of the numeric values of A and its factors as well as monitoring to determine m . Both of these measures can be performed without affecting the factorization modules and with little additional cost. The Wilkinson error bound requires monitoring of all values in each step of the elimination. Its order of complexity is the same as the factorization and must be performed simultaneously with the factorization. The computation of the actual error can be performed at cost similar to

the evaluation of the Wilkinson bound, without affecting the factorization. It also requires more storage.

In practice, only the first two estimates, finding the error in a specific solution or computing the Erisman-Reid bound, are likely to be low enough in cost. Since the MA28 code provides a built-in option to compute the product of the norms of L and U used in the Erisman-Reid bound, this bound is reasonable to use with this code. (However, in his complex version of MA28 Duff actually monitors the growth, as in (3); he argues that the overhead of complex arithmetic on some computers makes the monitoring inconsequential on these computers.) Estimating the error in a specific solution could be used with other codes, but it should be used with caution. The particular vector used here is not a good choice in general [19]. More reliability would be gained by using two or more vectors with random values.

4. Conclusions and recommendations for further study. We set out to determine whether any of the orderings in this study could be regarded as generally effective. The results clearly indicate that none of the orderings, considered strictly as preorderings, can be so regarded. However, we confirmed earlier findings that the stabilized version of the Markowitz heuristic, as represented by the MA28 code, is a generally useful and effective reordering.

Our study of orderings is incomplete in that it did not include stabilized versions of the P^4 and P^5 heuristics. As we indicated in the previous section, a stable and implicit P^5 factorization would be quite valuable, if it exists, although its range of applications would be restricted to certain problem classes. We are not sanguine about the likelihood of developing such a variant. A stable version of the P^4 heuristic is already known. The published experience with this algorithm is generally negative, but these results are also confined to a narrow class of problems, linear programming bases.

Our study is incomplete also in that fill and stability are not the only measures of effectiveness for codes; ordering and factorization times are also important. The P^4 heuristic is less complex than the Markowitz heuristic, so ordering times for a good implementation of P^4 should be much less than the ordering time for the Markowitz ordering. It is often the case that the ordering dominates the computation for MA28 factorizations. A stable P^4 ordering and factorization may have smaller total cost, if more fill, than its Markowitz counterpart when only a single system with a given structure is to be solved. Whether this is true will depend on how sensitive P^4 is to pivoting for stability, and whether the limited pivoting, "living dangerously," recommended in [26] is, in fact, sufficient to obtain stability. These questions should be addressed directly by further work on this algorithm.

Recently Michael Saunders [27] has developed a variation on the standard sparse Gaussian elimination that simplifies the modification of the factors of the sequence of basis matrices in linear programming problems. Saunders uses two by two stabilized elementary matrices to compute and update the factors, and obtains potentially quite different interchanges in computing stable factorizations. Stability is assured for the initial factorization, given P^4 or any other suggested ordering. The results of this study do not apply directly to his new algorithm, but it remains likely that P^4 would not perform well on structurally symmetric problems.

Our results support the conventional wisdom that there are classes of sparse matrices, which differ in difficulty. Some of the areas studied resulted in matrices that were difficult for all of the algorithms, and others resulted in matrices that were easy for all of the algorithms. Our statistical analyses provide some suggestive results, but the size and characteristics of our sample are too limited to do more than point the

way for further investigation. A larger sample would have statistical benefits; it would have a correspondingly large cost in computing. Care should be taken to provide a wide enough selection to control the effects of certain parameters. For example, structural symmetry appears to be an indicator of high fill, but it is clearly related in most of our sample to additional parameters such as regularity and relatively high density. This study presupposed the value of general purpose sparse matrix heuristics and codes. The existence of problems such as the relatively unstructured matrices in our test collection, the linear programming bases and the chemical engineering models, confirms this assumption. However, the highly structured and nearly symmetric problems, which proved the hardest for these general heuristics, point to the need for further study of the special cases where variations of symmetric orderings are appropriate. This becomes particularly true when stability can be incorporated into the factorization. This is not the case for most current orderings for nearly symmetric matrices, which assume that stability is not an issue. The recent work on extensions of frontal methods by Duff and Reid [8], [10]-[12] provides a possible basis for such methods. Such methods are quite different in character from the general sparse orderings, so careful analysis will be required in any comparison. Only with such comparison will we determine if the general sparse heuristics are still valuable for this special class of problems.

Note added in proof. An additional measure of estimating the error in Gaussian elimination without pivoting was reported by Chu and George after it could have been incorporated into this paper. We regret this omission and refer interested readers to the report [E. Chu and J. A. George, *An algorithm to estimate the error in Gaussian elimination without pivoting*, Faculty of Mathematics, University of Waterloo, Waterloo, Ontario, CS-84-21, 1984].

REFERENCES

- [1] J. BISSCHOP, Y. LEVY AND A. MEERAUS, *Programs for structure analysis of sparse matrices*, Tech. Report, Development Research Ctr., World Bank, Washington, DC, 1979, submitted to ACM Trans. Math. Software.
- [2] I. S. DUFF, *On permutations to block triangular form*, Bull. Inst. Math. Appl., 19 (1977), pp. 339-342.
- [3] I. S. DUFF AND J. K. REID, *Some Design Features of a Sparse Matrix Code*, AERE-Harwell Report CSS 48, 1977.
- [4] I. S. DUFF, MA28—*A set of FORTRAN subroutines for sparse unsymmetric linear equations*, AERE—R8730, Her Majesty's Stationery Office, London, 1977.
- [5] I. S. DUFF AND J. K. REID, *An implementation of Tarjan's algorithm for the block triangularization of a matrix*, ACM Trans. Math. Software, 4 (1978), pp. 137-147.
- [6] ———, *Algorithm 529: permutations to block triangular form*, ACM Trans. Math. Software, 4 (1978), pp. 189-192.
- [7] ———, *Performance evaluation of codes for sparse matrix problems*, in Performance Evaluation of Numerical Software, L. Fosdick, ed., North-Holland, New York, 1979, pp. 121-135.
- [8] I. S. DUFF, *Design features of a code for solving sparse unsymmetric linear systems out-of-core*, AERE-Harwell Report CSS 89, 1981.
- [9] I. S. DUFF, R. G. GRIMES, J. G. LEWIS AND W. G. POOLE, JR., *Sparse Matrix Test Problems*, SIGNUM Newsletter, 17, 2 (1982), p. 22.
- [10] I. S. DUFF AND J. K. REID, MA27—*A set of FORTRAN subroutines for solving sparse symmetric sets of linear equations*, AERE—R10533, Her Majesty's Stationery Office, London, 1982.
- [11] ———, *The multifrontal solution of indefinite sparse symmetric linear systems*, AERE-Harwell Report CSS 122, 1982.
- [12] ———, *The multifrontal solution of unsymmetric sets of linear equations*, AERE-Harwell Report CSS 133, 1983.
- [13] A. M. ERISMAN AND J. K. REID, *Monitoring the stability of the triangular factorization of a sparse matrix*, Numer. Math., 22 (1974), pp. 183-186.

- [14] A. M. ERISMAN, R. G. GRIMES, J. G. LEWIS AND W. G. POOLE, JR., *A structurally stable modification of Hellerman-Rarick's P^4 algorithm for reordering unsymmetric sparse matrices*, SIAM J. Numer. Anal., 22 (1985), pp. 369-385.
- [15] P. E. GILL, W. MURRAY, M. H. WRIGHT AND M. A. SAUNDERS, *Sparse matrix methods in optimization*, Tech. Rep. SOL 82-17, Dept. of Operations Research, Stanford Univ., 1982.
- [16] M. T. HEATH, *Some extensions of an algorithm for sparse linear least squares problems*, this Journal, 3 (1982), pp. 223-237.
- [17] E. HELLERMAN AND D. C. RARICK, *Reinversion with the preassigned pivot procedure*, Math. Programming, 1 (1971), pp. 195-216.
- [18] ———, *The Partitioned Preassigned Pivot Procedure (P^4)*, in *Sparse Matrices and Their Applications*, D. J. Rose and R. A. Willoughby, eds., Plenum Press, New York, 1972, pp. 67-76.
- [19] W. KAHAN, private communication.
- [20] T. D. LIN AND R. S. H. MAH, *Hierarchical partition—a new optimal pivoting algorithm*, Math. Programming, 12 (1977), pp. 260-278.
- [21] ———, *A sparse computation system for process design and simulation, part I. Data structures and processing techniques*, AIChE J., 24 (1978), pp. 830-839.
- [22] R. S. H. MAH AND T. D. LIN, *A sparse computation system for process design and simulation, part II. A performance evaluation based on the simulation of a natural gas liquefaction process*, AIChE J., 24 (1978), pp. 839-848.
- [23] H. M. MARKOWITZ, *The elimination form of the inverse and its application to linear programming*, Management Sci., 3 (1957), pp. 255-269.
- [24] B. A. MURTAGH AND M. A. SAUNDERS, *MINOS User's Guide*, Technical Report 77-9, Systems Optimization Laboratory, Dept. of Operations Research, Stanford Univ., 1977.
- [25] N. H. NIE, C. H. HULL, J. G. JENKINS, K. STEINBRENNER AND D. H. BENT, *Statistical Package for the Social Sciences (SPSS)*, McGraw-Hill, New York, 1975.
- [26] M. A. SAUNDERS, *A fast, stable implementation of the simplex method using Bartels-Golub updating*, in *Sparse Matrix Computations*, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976, pp. 213-226.
- [27] ———, private communication.
- [28] J. K. REID, *A note on the stability of Gaussian elimination*, Bull. Inst. Math. Appl., 8 (1971), pp. 374-375.
- [29] R. E. TARJAN, *Depth first search and linear graph algorithms*, SIAM J. Comput., 1 (1972), pp. 146-160.