Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

M. Louter-Nool, D.T. Winter

Benchmark of the initial release of the LAPACK library

Department of Numerical Mathematics Note NM-N8903 November

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

Benchmark of the initial release of the LAPACK library

Margreet Louter-Nool, Dik T. Winter
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

This Note reports on the test results of a pre-release of LAPACK, a new scientific library for solving linear equations, eigenvalue problems and linear least squares. Results for five different high-performance computers have been collected in graphs. The figures have been arranged in such a way that each page contains the results for one particular routine for all tested machines. It can be easily verified whether characteristics like the performance ratio for blocked and unblocked routines, or for lower or upper triangular storage schemes are machine dependent or not. Some hints are given to improve the timing procedure, on the one hand to simplify the process of timing, on the other hand to modify the values of the input parameters. In conclusion, critical notes on the implementation, the contents and the selected datastructure of this release are given.

1980 Mathematics subject classification: Primary:65V05. Secondary:65FXX
Key Words & Phrases: Level 1, 2, 3 BLAS, Vector and Parallel Computers, Efficiency, Portability, Performance Measurements.

1. Introduction

A new scientific library called LAPACK is developed by Argonne National Laboratory, in conjunction with the Courant Institute and the Numerical Algorithms Group, Ltd. This transportable library in FORTRAN77 is based on the well-known LINPACK[3] and EISPACK[7,13] packages for solving linear equations, eigenvalue problems and linear least squares. We at CWI have been invited to serve as a test site for this project, and to comment on the contents and on the current implementation of these routines.

Currently CWI has access to quite a few vector and parallel machines and in consultation with Argonne we have decided to run the codes on five machines: Alliant FX/4, IBM 3090-VF, CDC CYBER 995, CDC CYBER 205, and NEC SX-2. The first release of LAPACK[1] contains only a fraction of the routines that will ultimately be part of LAPACK. In spite of this limited size, hundreds of tables were generated to measure efficiency. In order to be able to interpret these values we decided it to be helpful to plot these values. The figures have been arranged in such a way that each page contains the results for one particular routine on several machines. At one glance it can be seen whether the blocked version of a routine has to be preferred or not and how many right-hand side vectors are required to achieve high performances. It can also be easily verified whether such characteristics are machine dependent or not.

We intend to give the reader a global insight in the performance of this LAPACK release. In Appendix A, we list the subroutines which are included in the initial release. This Note reports on how we have carried out the tests and performance measurements. Also some hints are given to improve the timing procedure. The problems we encountered during the timing and testing are outlined in Winter[14]. Finally, we make some critical remarks concerning the present LAPACK prerelease.

Note NM-N8903
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

FX4:	Machine:	Alliant FX/4	
	OS:	Concentrix V4.0	
	Compiler:	FORTRAN V4.0	
	Options:	-O -DAS -c -g	
	BLAS:	Level 1, 2	- system supported
		DGEMM, SGEMM (Level 3)	- system supported
		rest of Level 3	- portable FORTRAN
205:	Machine:	CDC CYBER 205-642	
	OS:	VSOS 2.3 SYS 690C	
	Compiler:	FORTRAN 200 CYCLE 690B	
	BLAS:	portable FORTRAN	
	Options:	OPTIMIZE, UNSAFE	
205.opt:	Machine:	CDC CYBER 205-642	
	OS:	VSOS 2.3 SYS 690C	
	Compiler:	FORTRAN 200 CYCLE 690B	
	BLAS:	Level 1, 2	- optimized codes[9,10,11]
		Level 3	- portable FORTRAN
	Options:	OPTIMIZE, UNSAFE	
995:	Machine:	CDC CYBER 995E	
	OS:	NOS/VE 1.4.1 L716AE	
	Compiler:	VFTN V2.3	
	BLAS:	portable FORTRAN	
	Options:	OPTIMIZATION_LEVEL=HIGH	
		VECTORIZATION_LEVEL=HIGH	
NEC:	Machine:	NEC SX2	
	OS:	SXOS R3.11	
	Compiler:	FORTRAN77/SX REV. 039	
	BLAS:	portable FORTRAN	
	Options:	none	
IBM:	Machine:	IBM 3090-180VF	
	OS:	VM/HPO 4.2 level 8803,	
		CMS L1 4.2/8803	
	Compiler:	VS FORTRAN LEVEL 2.3.0	
	BLAS:	portable FORTRAN	
	Options:	OPTIMIZE(3) VECTOR	

TABLE 1 Review of the Machines in the Benchmark

2. THE MACHINES

In Table 1 we review the machines used in this benchmark. For convenience, we refer to the machines by the names listed in the first column. The operating systems, the compilers and the compiler options used are shown, too. In Appendix B we give a review of our testing and timing of LAPACK. Some timing sets have not completely carried out on all available machines. This appendix gives possible explanations where tests failed or timings were aborted. We encountered, among others, the following problems:

- a few (undocumented) machine dependencies occur
- a few non standard constructs are used (admittedly in nonstandard routines)
- compiler problems
- the sources supplied made a lot of assumptions about the target system, as if it were a UNIX™ system, which is not true for the CYBERs, the NEC and the IBM machine in this benchmark.

These problems are discussed in a separate Note on porting[14] of this LAPACK release.

Most routines in LAPACK occur in four versions : REAL, DOUBLE PRECISION, COMPLEX and DOUBLE COMPLEX. On several machines we indeed have timing results of all four precisions. However, we have decided to consider the results of only one comparable (64-bit floating point) precision per machine, this implies REAL on the 205 and 995 and DOUBLE PRECISION on the IBM, NEC and FX4 and, if available (see Appendix B), one COMPLEX precision.

The performances of LAPACK depend heavily on the implementation of the BLAS (Basic Linear Algebra Subprograms[4,5,8]). Actually, it is not fair to present LAPACK results without mentioning the performances of BLAS. However, in the best case, the BLAS is system supported and then the performances are often difficult to explain. In the worst case, the portable unoptimized implementations had to be used. For more details on the performance of the portable Level 2 BLAS, we refer to Louter-Nool[12]. Here, we restrict ourselves to review which BLAS have been used. Although on the FX4 all levels of BLAS exist, only Level 1 and 2 BLAS and two Level 3 BLAS routines have been used in this LAPACK benchmark. None of the Level 3 BLAS routines passed the error tests, except for SGEMM and DGEMM, which both compute the general matrix-matrix product. Since the CYBER CDC 205 is not able to vectorize most of the BLAS routines written in portable FORTRAN well, and since the optimized BLAS implementations[9,10,11] are probably not commonly used, we produced timing results with and without these optimized BLAS implementations (cf. the results for 205 and 205.opt). On all other machines portable FORTRAN implementations have been used.

3. EXPERIMENTAL RESULTS

3.1. Timing procedure

By the timing program's input file, one can control the size of the test matrices, the blocksize for blocked routines, the bandwidth for banded matrices, the leading dimension for the work arrays and the individual routines to be timed (see also Anderson and Dongarra[1]). The actual input files contain the following values :

5					Number of values of n (the order)
32	64	128	256	512	S_n: the values of n
5					Number of values of *nb* (the blocksize)
1	2	8	16	32	S_{nb}: the values of *nb*
5					Number of values of k (the bandwidth)
31	63	127	255	511	S_k: the values of k
2					Number of values of *lda* (the leading dimension)
512	513				S_{lda}: the values of *lda*
SGE	TRF	TRS	TRI	CON	
SGB	TRF	TRS		CON	
SPO	TRF	TRS	TRI	CON	
SPP	TRF	TRS	TRI	CON	
SPB	TRF	TRS		CON	
SSY	TRF	TRS	TRI	CON	S_{path}: the LAPACK path names
SSP	TRF	TRS	TRI	CON	
SSB	TRF	TRS		CON	
SGE	QRF	QRS			
STR			TRI		
STP			TRI		

TABLE 2 Parameter Values of the Timing Procedure

For the blocked routines a blocksize of 1 means that the unblocked version is called. The general outline of the timing procedure is as follows (not all options are applicable for each path name) :

```
for each relevant value of UPLO
  for LDA ∈ S_lda
    for K ∈ S_k
      for each path name ∈ S_path
        for N ∈ S_n
          for NB ∈ S_nb
            Time the routines in this path
```

The parameter UPLO specifies whether the upper or lower triangular part of the matrix is stored.

In Appendix A, Table 1, a review is given of all routines involved in this release. The routines operating on unitary and orthogonal matrices (i.e., _UNGNC, _UNMLC, _ORGNC and _ORMLC) have been added to the present release, but, unfortunately, both the main test program and the main timing program of this initial distribution of LAPACK[1] does not recognize these routines. A still considerable number of 38 routines has been timed, some of them for several bandwidths and for either upper or lower triangular storage. We mention that we have also tested and timed the Level 2 and 3 BLAS completely, but these results will not be reported here.

The original output of the timing programs is in the form of tables which show megaflop rates for each routine over all values of N and NB. Apparently, the difference in performance between LDA=512 and LDA=513 is very small. All figures represent results for LDA=512.

3.2. Remarks on this benchmark

- Only results of LAPACK on shared memory machines are considered in this Note. This new library should also be suited for computers with other hierarchies of memory, such as cache, local memory or vector registers, and for parallel processing computers. For those architectures it is often preferable to partition the matrix or matrices into blocks and to perform the computation by matrix-matrix operations on the blocks. Though many LAPACK routines operate on blocks the matrices are not explicitly required to be stored blockwise; the matrices are supposed to be stored in the usual FORTRAN way (i.e., columnwise). Moreover, the dimensions of the submatrices - on which LAPACK operates - are kept as large as possible (i.e., multiples of the blocksize).
- No optimal Level 3 BLAS implementations were used, except for DGEMM and SGEMM on the FX4. This explains, that on machines with optimized Level 2 BLAS, performances of blocked routines using Level 3 BLAS can be disappointingly compared with performance of unblocked routines which exploit (optimized) Level 2 BLAS.

3.3. Presentation of the results

We have preferred to present the results in graphs rather than tables. For each routine eight figures, displaying the Megaflop rates for five different N values, have been plotted, namely six figures for the REAL case and two for the COMPLEX case. In all figures a dotted line indicates that a blocked version has been used. In that case the markers specify for which blocksize the performance is obtained. For the solve routines, which solve the system A . X = B, the matrix B can contain four different numbers of right-hand sides : 1, 2, ½ N, and N. We have tried to combine the results as much as possible. As an illustration, if performance turns out to be independent of the number of right-hand sides, then the results of all bandwidths involved for such routine have been represented by one curve only. In that case the markers specify the bandwidths and not the number of right-hand sides. If possible, the results of upper triangular and lower triangular storage are shown in the same figure. We mention that we have completely tested and timed the class of routines indicated by __CON, which compute

the condition number. However, we have not supplied any of its results, since most of these routines execute at scalar speed (and therefore produce figures almost coinciding with the x-axis).

The LAPACK timing procedure LAOPS computes incorrectly the number of floating point operations for _POTRI and _PPTRI, and hence their megaflop rates. The true values for these routines are twice as high. In order to show the right performances the scaling in the relevant graphs has been adapted. Furthermore, we mention that a complex multiplication takes 6 floating point operations rather than 4 as is used in the codes. Unfortunately, we only noticed this after completing the experiments. Evidently, the total number of floating point operations for a COMPLEX routine is specified by both the number of multiplications and the number of additions. Therefore, it is not sufficient to increase the Mflop rates with a scaling factor of 6 / 4. It would have been too time consuming to correct these values and, consequently, they are still incorrect.

3.4. Short discussion on results for each subroutine

_GETRF The blocked version (the dotted lines in figure 1a-h) performs very badly on both the 205 and the 205.opt. On the FX4, the 995 and the NEC, the performance of the blocked version is comparable to those of the unblocked one (solid lines) for small and large matrices. For the IBM the unblocked version is superior for N from 128 up to 512.

_GETRS The 205 results are very poor. Only for the NEC, the number of right-hand sides makes a difference.

_GETRI The unblocked version performs badly on the 205 while it is favorite on the 205.opt. On the 995 the blocked routine scores very well for all blocksizes.

_GBTRF The results of three different bandwidths have been collected in one figure. Performances increase, if the bandwidths increase. The results for the NEC are disappointing; for a bandwidth of 31, the performances do not exceed 20 Mflops. Note that performance decreases for $N > 128$ on the FX4.

_GBTRS Analogously to _GBTRF we have gathered the results for all bandwidths in one figure. Four different numbers of right-hand sides were used in the timing procedure. As is shown, this number hardly influences the performance.

_POTRF Two versions for the factorization of a positive definite matrix have been incorporated in this release, one for matrices stored in upper triangular form and one for matrices stored in lower triangular form. In LINPACK only lower triangular storage is accepted. It turns out that for all machines the routine which operates on the lower triangular form is to be preferred both in the block and unblocked case. The unblocked implementation is superior in that case. The behavior of the routine for upper triangular storage is rather unpredictable.

_POTRS Performance for the upper and lower triangular storage cases are identical, except for the 995; its upper triangular storage performance figure is very surprising. The _POTRS implementation is not very useful for the 205 and the 205.opt; it does not exceed scalar speed.

_POTRI Similar to _POTRF the results strongly depend on whether matrices are stored in lower or upper triangular form. Note that here, as opposed to the corresponding factorization routine _POTRF, the upper triangular storage scheme is favorite. The blocked version also scores very well for the upper storage case.

_PPTRF For the FX4, the 995 and the IBM it does not matter which storage scheme is chosen. For the NEC and the 205.opt the lower triangular storage scheme performs much better, even better than the non-packed routine _POTRF. Notice that the 205 does not vectorize the packed storage scheme.

_PPTRS Performances obtained by _PPTRS are much higher than those obtained by _POTRS for the IBM and the 205.opt. Probably the implementation of the unpacked version _POTRS can be improved.

_PPTRI The results are comparable to _POTRI and _PPTRS.

_PBTRF UPPER : For dense matrices (see Figure 20a-h with $K=511$) performance resembles the _POTRF UPPER results. For banded matrices with less than 25% nonzero elements ($K=32,64$)

this routine seems to be useless.
LOWER : For the lower triangular case performances for small bandwidths are not as bad as for the upper triangular case for the "small" systems. However, for the FX4 the difference in performance for _POTRF and _PBTRF is so big that the use of _POTRF has to be preferred even for banded matrices thereby ignoring all zero elements. Notice that for the FX4, the complex case slightly differs from the real case.

_PBTRS As compared with the results of _POTRS for dense matrices, much higher performance is obtained, especially for the 205.opt and the IBM. The results for upper triangular and lower triangular storage are exactly the same.

_SYTRF On the FX4, performance for the real case does not exceed 5 Mflops; on all of the other machines performance still increases with the order of the matrix. The results for lower and upper storage are almost identical. For the complex case, the exceptionally bad performances can be explained by the absence of system supported Level 2 BLAS routines for complex symmetric matrices. Unfortunately, the set of Level 2 BLAS does not provide for this matrix type.

_SYTRS In particular for the 205, the 205.opt and the NEC the number of right-hand sides is very important for this solve routine. For one or two right-hand side vectors this routine does not achieve high megaflop rates. Both versions on the 995 behave rather silly for N right-hand side vectors(Fig.29b,Fig.30b).

_SYTRI On the 205, the 205.opt, the NEC and the 995 the results for the invert routine _SYTRI are worse than those for the solve routine _SYTRS with N right-hand sides. This implies that _SYTRS which solves a linear system A.X=B of, for example, N equations with B arbitrary (including identity) is faster than _SYTRI which only computes the inverse.

_SPTRF The packed storage version of _SYTRF displays the same characteristics as the unpacked storage version, except for the 995 and the 205. The latter machine can not vectorize packed storage mode. For ZSPTRF on the FX4, see _SYTRF.

_SPTRS There is no difference in performance between the routine for packed matrices and the one for matrices stored in the usual way. No BLAS routines for packed matrices are used, so even for the 205 the performance is high.

_SPTRI see _SPTRF.

_SBTRF An error message is printed for K=511, so only four different lines are drawn in figures 36a-h. For bandwidths up to 255 the results for matrices of full order correspond to the results of the dense matrix routine _SYTRF. Again for K=31,63 the results are poor. When comparing the upper and lower triangular storage results one can conclude that they are nearly equal.

_SBTRS Similar to the case for non-banded matrices, the influence of the number of right-hand sides is very significant. As one can expect, the bandwidth must also be taken into account. Its influence on the 205, the 205.opt and the NEC is larger than on other machines. On the 995, performance decreases for N from 128 to 256 when the number of right-hand sides is equal to N. This happens for all bandwidths. A similar decrement appears for the FX4, but less significant.

_GEQRF Results were obtained for three different values of N, namely N=⅓M, ⅔M, M, where the matrix is of dimension M × N. The difference in performance is not striking, especially for the unblocked case. On the 205, the 205.opt and the NEC the results of the blocked version approximate those of the unblocked one. For blocksizes of 32 and 64 and for M=256, 512 the blocked version has to be preferred for the FX4 as opposed to the IBM on which the blocked version performs badly.

_GEQRS Analogously to _GEQRF, three values for N are chosen. Obviously, the present implementation is not particularly suited for the 205 and the 205.opt, whereas it performs well for the NEC when the number of right-hand side vectors is big enough.

_TRTRI The upper and lower storage cases perform largely the same, except for the unblocked

routine on the 995, the 205 and the 205.opt. On the 995 and the 205 the unblocked, lower storage results are much worse than the blocked ones, whereas the 205.opt results are much better.

_TPTRI The difference in performance for the upper and lower storage mode is less significant than for the unblocked version of _TRTRI. Again we see that the 205 does not vectorize the packed version.

Routines for Hermitian matrices

Figures 55-72 show the performance of the routines for Hermitian matrices. The graphs can be compared with those for the symmetric complex matrices in Figures 22-45d,h. It is spectacular how completely different the graphs for the complex symmetric factorization and invert routine are, compared with the Hermitian case for the FX4. The difference can be explained by the absence of system supported BLAS2 routines for symmetric complex matrices. On the NEC only portable implementations were used and it appears that operations on both symmetric complex and Hermitian matrices are equally well optimized.

4. Remarks and conclusions

We support the design and implementation of LAPACK, and we believe that it will be a good successor of LINPACK and EISPACK. As a test site of the project we discovered how comparatively simple it was to execute the codes, but we realize how difficult it will be to analyze the results of all the test sites. In general, no big problems appeared during the extensive testing and timing of both LAPACK and BLAS. The sources are well readable and documented. Besides, we remark that also the output files are very easy to interpret. It is clear that the obtained results are a good benchmark not only for LAPACK but for the machines as well.

In the next subsections we mention some wishes concerning the timing procedure. Finally, we comment on the efficiency of this LAPACK release.

4.1. Wishes with respect to the timing procedure

For the next release we suggest the following modifications concerning the timing procedure (see Section 3.1):

- Bandwidths and blocksizes ought to correspond to the order of the matrix.
  - Exactly 60% of the banded matrices in the present timing procedure are of full order. We prefer to include performance measurements of really banded matrices, for example, with 2, 3, 5, or 8 diagonals.
  - Blocksizes should be much smaller than the order of the matrix; blocksizes greater than the order of the matrix are not realistic. The experiments indicate that blocksizes 2 (and 8) should be replaced by larger values, for example, 16 or 24.
- A mechanism ought to be developed whereby the time measuring routines can determine good values for N, NB and LDA.
  - It is not useful to time for large values of N when a routine is executing at scalar speed. It is also possible that cache effects dominate to such an extent that it is better to use smaller values for N.
  - Until it is clear that the leading dimension LDA influences the performance, only one LDA value suffices in the timing procedure.
- More restarting points are needed for timing. In this release a routine can only be timed as prescribed by the timing procedure. If for some reason (e.g., the machine goes down or a compiler bug) the timing procedure stops, then it is not possible to restart with, for example, UPLO equal to LOWER. Therefore, the timing procedure should generate an output file which contains information about the current run. By using that output file it should be possible to restart before or just after the last obtained value. For more details the reader is referred to Winter[14].
- All of the N values specified are powers of 2. It would be interesting to consider odd numbers as

well. Besides that, blocksizes which are no divisor of N should be specified, too.

4.2. Some critical notes on the implementation

We think that it is necessary to investigate other data structures, too. This LAPACK release allows symmetric, Hermitian and triangular matrices to be stored in either upper or lower triangular matrices. Moreover, routines for the out-of-date packed storage mode have been supplied. But why are other data structures not included?

First of all, we mention the "block" form, in which the original matrices are partitioned into submatrices or blocks, and the algorithms are expressed in terms of basic matrix-matrix operations on the blocks, as was planned in the first working note of this project by Demmel et al.[2]. As is shown in Section 3.3, for most routines the unblocked or Level 2 BLAS implementation has still to be preferred. To take full advantage of the blocked version, the matrix must be stored blockwise. By operating on small blocks, the amount of data movement can be limited. Besides, if matrices were partitioned into blocks explicitly, the operations on distinct blocks could be performed in parallel. Obviously, this approach requires optimal BLAS implementations for small blocks for one single processor and, of course, directives to indicate which parts can operate in parallel. Currently, on the FX4, the maximum performance for BLAS routines is obtained for large matrices using all available processors. For matrices of smaller order, for example, 32, only low megaflop rates are realized. At the moment, the only possible way to exploit parallelism is to operate in parallel at loop level within the BLAS kernels. In this case the blocksize should not be too small.

Secondly, we prefer to allow another data structure for banded matrices apart from the present data structure. The prescribed data storage requires diagonals to be stored in rows accordingly to the Level 2 BLAS storage mode. If diagonals were stored in columns then longer contiguous vectors are obtained and the effect of vectorization will increase. Again this benchmark shows that, at least for several architectures, the present storage mode is inadequate for small bandwidths. Our proposal implies that also the current set of BLAS needs to be extended. As a side effect, routines for solving bi- and tri-diagonal systems, not yet available in LAPACK, become a part of the set of BLAS. Moreover, since the set of Level 2 BLAS does not provide for routines for complex symmetric arithmetic, five pseudo BLAS2 routines have been added as auxiliary routines of LAPACK. In our opinion the set of BLAS2 must be permanently extended with these routines.

In Louter-Nool[12], LINPACK routines based on Level 2 BLAS are discussed. In LINPACK implementations, calls to Level 1 BLAS were replaced by calls to Level 2 BLAS without changing the algorithm, the data structure and the round-off pattern. As expected, large speedups were obtained. Under the same conditions as for this LAPACK release, matrices of order 255 were time-measured on the CDC CYBERs 995 and 205 (including 205.opt timings) and the NEC SX2. Megaflop rates of the original and the modified LINPACK have been compared. It turned out that for general matrices much higher performance could be achieved than for the corresponding LAPACK routines. A brief inspection of the LAPACK routines for factorization shows that, for example, row-interchanging is not performed by a simple _SWAP on a complete row, but in each column separately, which prohibits vectorization (cf. SGEFA implementation in [12, Section 3.1]). The computation of the inverse of a triangular system _TRTRI can be improved, too. The modified LINPACK megaflop rates are about twice as high as the for LAPACK.

We anticipated LAPACK to perform much better than (modified) LINPACK, since new algorithms could have been implemented, and parameter lists could have been adapted as well. As an illustration, we mention that LINPACK allows only one single right-hand side vector, whereas LAPACK can solve more systems simultaneously. For some of the solve routines this results in performance improvements, for others it does not. Moreover, it occurs that invert routines are less efficient than the corresponding solve routine. Summarising, at least some of the routines of this LAPACK release can be improved and other data structures must be considered.

REFERENCES

1. EDWARD ANDERSON and JACK DONGARRA (April 1989). LAPACK Working Note #10, Installing and Testing the Initial Distribution of LAPACK, *Technical Memorandum No. MCS-TM-130, Argonne National Laboratory*.
2. J. DEMMEL, J.J. DONGARRA, J.J. DU CROZ, A. GREENBAUM, S. HAMMERLING and D.C. SORENSON (September 1987). Prospectus for the Development of a Linear Algebra Library for High-Performance Computers, *Technical Memorandum 97, Argonne National Laboratory*.
3. J.J. DONGARRA, J.R. BUNCH, C.B. MOLER and G.W. STEWART (1979). *Linpack User's Guide*, SIAM, Philadelphia, PA.
4. J.J. DONGARRA, J.J. DU CROZ, I.S. DUFF and S. HAMMERLING (May 1988). A Set of Level 3 Basic Linear Algebra Subprograms, *Technical Memorandum 88 (Revision 1), Argonne National Laboratory*.
5. J.J. DONGARRA, J.J. DU CROZ, S. HAMMERLING and R.J. HANSON (March 1988). An Extended Set of FORTRAN Basic Linear Algebra Subprograms, *ACM Transactions on Mathematical Software*, Vol. 14, No. 2, 1-17.
6. R.M. DUBASH, J.L. FREDIN and O.G. JOHNSON (1987). *Benchmark of the Extended Basic Linear Algebra Subprograms on the NEC SX-2 Supercomputer*, Lecture Notes in Computer Science, 297, Springer-Verlag, Berlin.
7. B.S. GARBOW, J.M. BOYLE, J.J. DONGARRA and C.B. MOLER (1977). *Matrix Eigensystem Routines - EISPACK Guide Extension*, Lecture Notes in Computer Science, 51, Springer-Verlag, Berlin.
8. C.L. LAWSON, R.J. HANSON, D.R. KINCAID and F.T. KROGH (1979). Basic Linear Algebra Subprograms for FORTRAN usage, *ACM Transactions on Mathematical Software*, 5, 308-323.
9. W.M. LIOEN, M. LOUTER-NOOL and H.J.J. TE RIELE (1987). Optimization of the real Level 2 BLAS on the CYBER 205, In: *Algorithms and Applications on Vector and Parallel Computers*, H.J.J. te Riele, Th. J. Dekker and H.A. van der Vorst (eds.), North-Holland, Amsterdam-New York-Oxford, 199-212.
10. M. LOUTER-NOOL (1987). Basic linear algebra subprograms (BLAS) on the CDC CYBER 205, *Parallel Computing*, 4, 143-165.
11. M. LOUTER-NOOL (June 1988). ALGORITHM 663, Translation of Algorithm 539: Basic Linear Algebra Subprograms for FORTRAN Usage in FORTRAN 200 for the CDC CYBER 205, *ACM Transactions on Mathematical Software*, Vol. 14, No. 2, 177-195.
12. M. LOUTER-NOOL (1989). LINPACK Routines Based on Level 2 BLAS, *The Journal of Supercomputing*, 3, 331-349.
13. B.T. SMITH, J.M. BOYLE, J.J. DONGARRA, B.S. GARBOW, Y. IKEBE, V.C. KLEMA and C.B. MOLER (1976). *Matrix Eigensystem Routines - EISPACK Guide*, Lecture Notes in Computer Science, 6, 2nd edition, Springer-Verlag, Berlin.
14. D.T. WINTER. Porting large packages of numerical software (to appear).

Appendix A. The LAPACK subroutine scheme

In this Appendix, we provide the subroutine scheme for LAPACK as described in Anderson and Dongarra[1] and indicate by means of a table which subroutines are included in this initial release.

Each routine name in LAPACK is a coded specification of the computation done by the subroutine. All names consist of six characters in the form TXXYYY. The first letter, T, indicates the matrix data type as follows :

S - REAL
D - DOUBLE PRECISION
C - COMPLEX
Z - COMPLEX*16 or DOUBLE COMPLEX

The next two letters, XX, indicate the type of matrix. In this LAPACK release, subroutines covering only a subset of the total collection of matrix types to be provided in LAPACK have been included. Most of these two-letter codes apply to both real and complex routines; a few apply specifically to one or the other, as indicated below :

GE - general (i.e., unsymmetric, in some cases rectangular)
GB - general band
PO - symmetric or Hermitian positive definite
PP - symmetric or Hermitian positive definite, packed storage
PB - symmetric or Hermitian positive definite band
SY - symmetric (i.e., indefinite)
SP - symmetric, packed storage
SB - symmetric band
HE - (complex) Hermitian (i.e., indefinite)
HP - (complex) Hermitian, packed storage
HB - (complex) Hermitian band
OR - (real) orthogonal
UN - (complex) unitary
TR - triangular
TP - triangular, packed storage

The last three characters, YYY, indicate the computation done by a particular subroutine. Included in this release are subroutines to perform the following computations :

TRF - perform a triangular factorization (LU, Cholesky, etc.)
TF2 - unblocked triangular factorization, if TRF is blocked
TRS - solve systems of linear equations (based on triangular factorization)
TRI - compute inverse (based on triangular factorization)
TI2 - unblocked computation of inverse, if TRI is blocked
CON - estimate condition number
QRF - perform the QR factorization without pivoting
QR2 - unblocked version of QRF
QRS - solve linear least squares problems (based on QR factorization)
GNC - generate a real orthogonal or complex unitary matrix as a product of Householder matrices, where each Householder vector is stored in a column of the matrix
GC2 - unblocked version of GNC
MLC - multiply a matrix by a real orthogonal or complex unitary matrix by applying a

product of Householder matrices, where each Householder vector is stored in a column of the matrix

MC2 - unblocked version of MLC

Given the above definitions, the following table indicates the LAPACK subroutines provided in this release :

	GE	GB	PO	PP	PB	HE SY	HP SP	HB SB	UN OR	TR	TP
TRF	×	×	×	×	×	×	×	×			
TF2	×		×		×						
TRS	×	×	×	×	×	×	×	×			
TRI	×		×	×		×	×			×	×
TI2										×	
CON	×	×	×	×	×	×	×	×			
QRF	×										
QR2	×										
QRS	×										
GNC									×		
GC2									×		
MLC									×		
MC2									×		

Appendix B. Timing tests on various types

In this Appendix we explain why the timing tests were done only for part of the types available on the different machines.

Alliant FX/4:
The timings for complex have not been completed. After more than 40 hours the machine went down. No significant differences between single and double precision complex appeared. The run for double precision stopped when the performance for general band matrices is checked. A restart with only LDA = 512 delivered no more problems.

CDC CYBER 205: On this machines, timings were only carried out for (64-bit) single precision and only in part for complex and double precision. For single precision only the case LDA=512 was done. The reason is that complex and double precision does not vectorize and that there is no timing difference between LDA=512 and LDA=513.

CDC CYBER 995: Also on this machine complex and double precision do not vectorize, so only timing data for (64-bit) single precision has been obtained.

NEC SX2: On this machine single and double precision real and complex do vectorize, but quadruple precision does not, so no timing was done for the latter case.

IBM 3090: The situation is similar to the NEC with respect to vectorization, but there are severe problems with complex arithmetic. The test programs that check for correct installation gave many errors, and therefore no attempt was made to do timings. The double precision timings were only carried out for matrices of order up to 256 because of budget constraints.

FIGURES 1 - 72

Graphs of the LAPACK Timing Results.

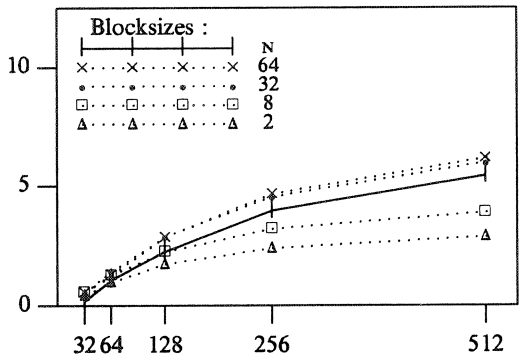

Fig.1a, FX4 : Timing data for DGETRF

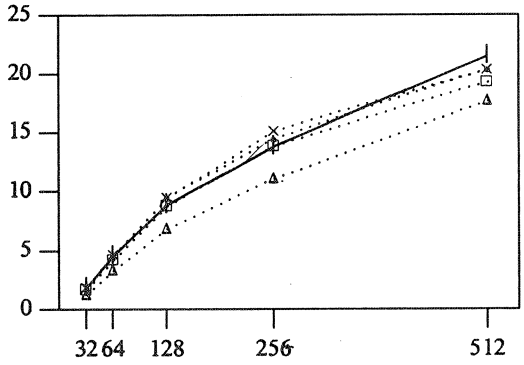

Fig.1b, 995 : Timing data for SGETRF

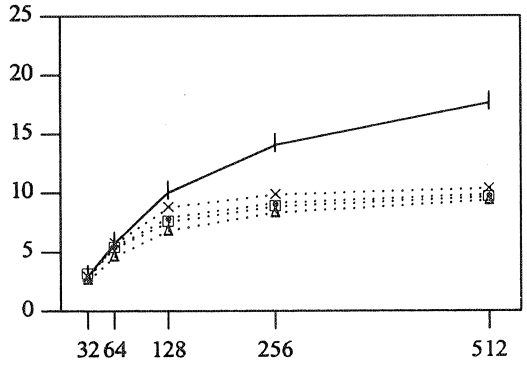

Fig.1c, IBM : Timing data for DGETRF

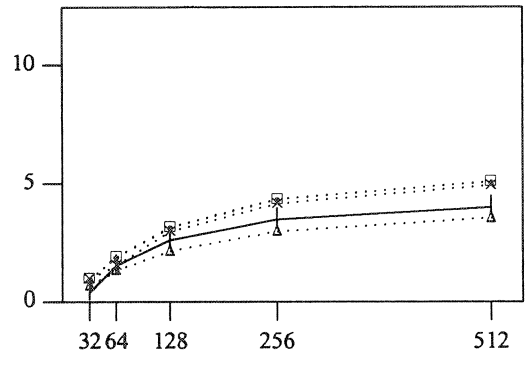

Fig.1d, FX4 : Timing data for ZGETRF

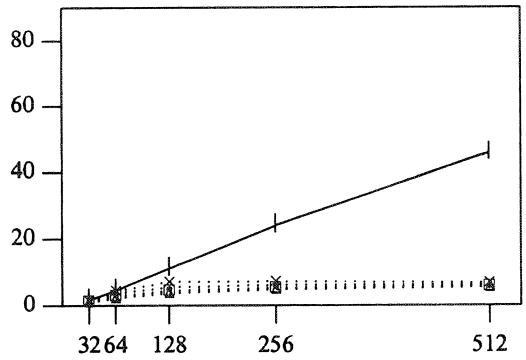

Fig.1e, 205 : Timing data for SGETRF

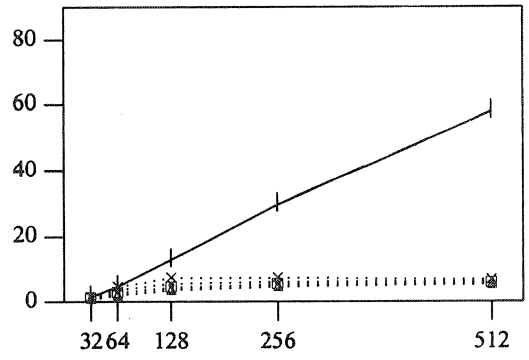

Fig.1f, 205.opt : Timing data for SGETRF

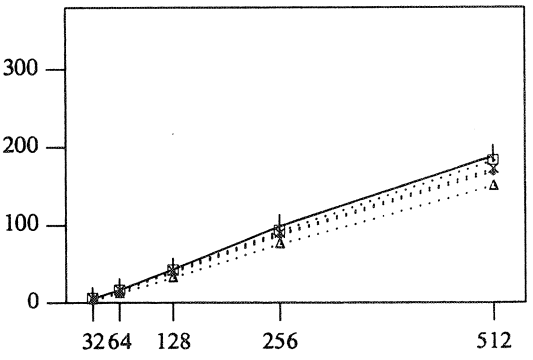

Fig.1g, NEC : Timing data for DGETRF

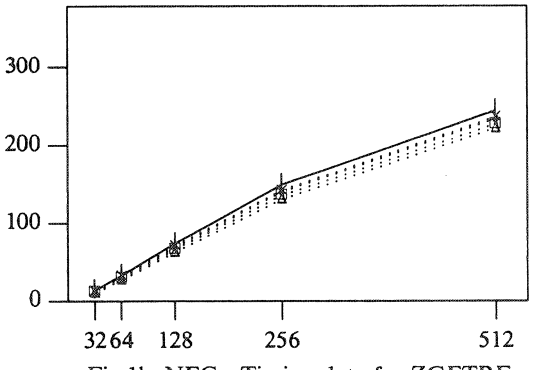

Fig.1h, NEC : Timing data for ZGETRF

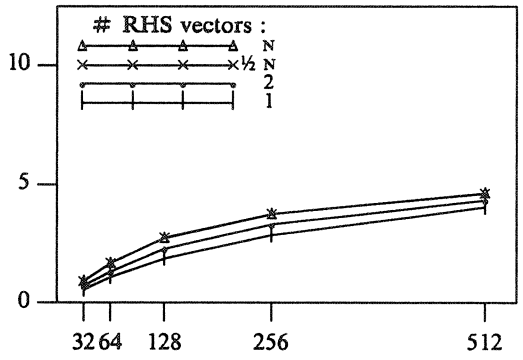

Fig.2a, FX4 : Timing data for DGETRS

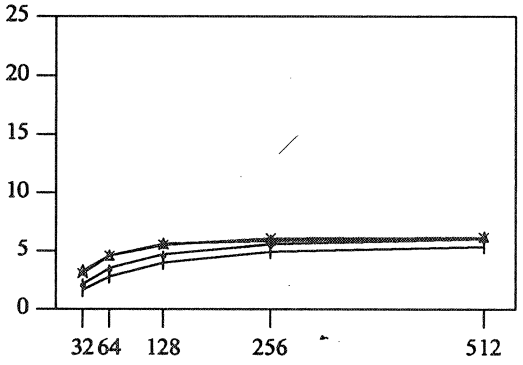

Fig.2b, 995 : Timing data for SGETRS

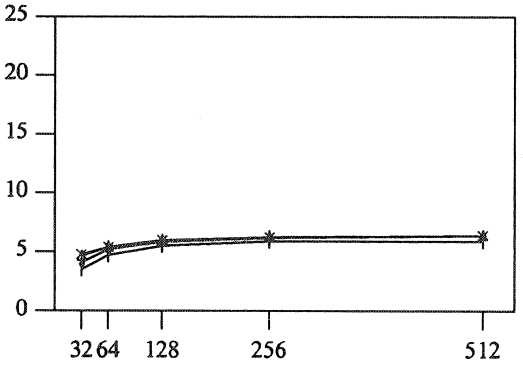

Fig.2c, IBM : Timing data for DGETRS

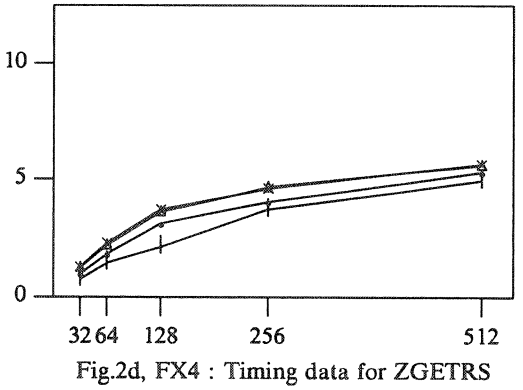

Fig.2d, FX4 : Timing data for ZGETRS

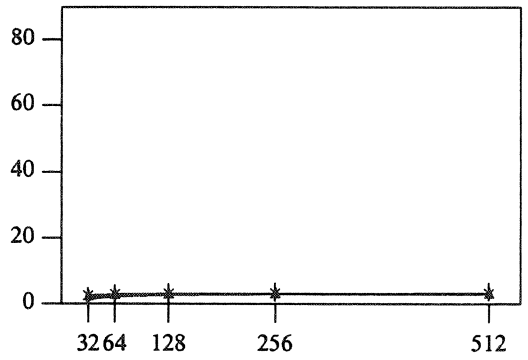

Fig.2e, 205 : Timing data for SGETRS

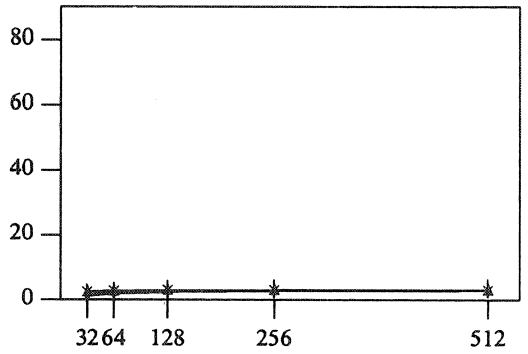

Fig.2f, 205.opt : Timing data for SGETRS

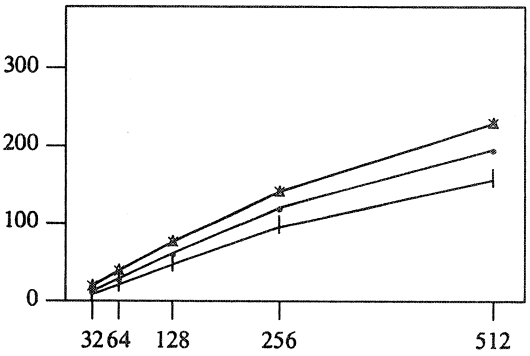

Fig.2g, NEC : Timing data for DGETRS

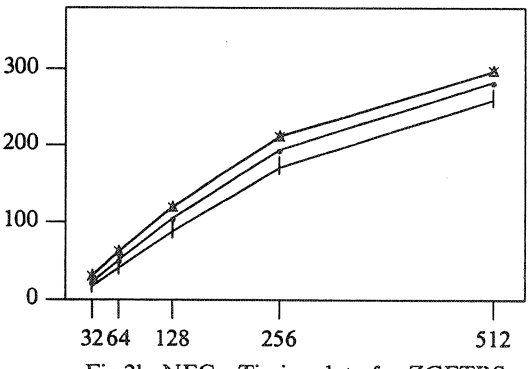

Fig.2h, NEC : Timing data for ZGETRS

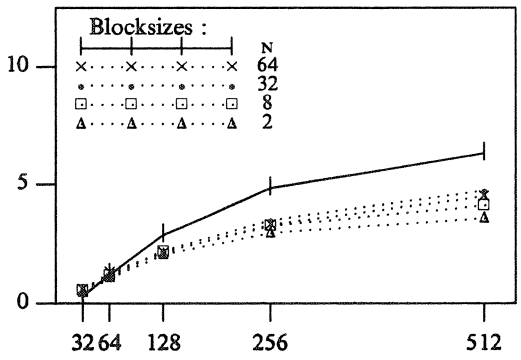

Fig.3a, FX4 : Timing data for DGETRI

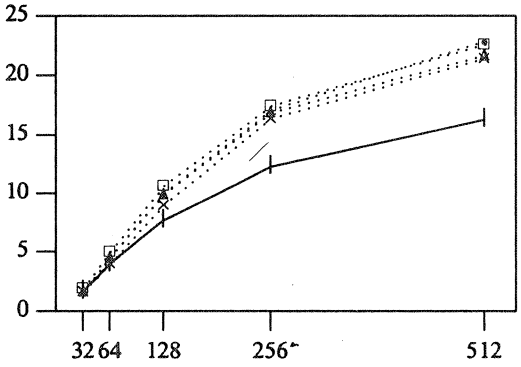

Fig.3b, 995 : Timing data for SGETRI

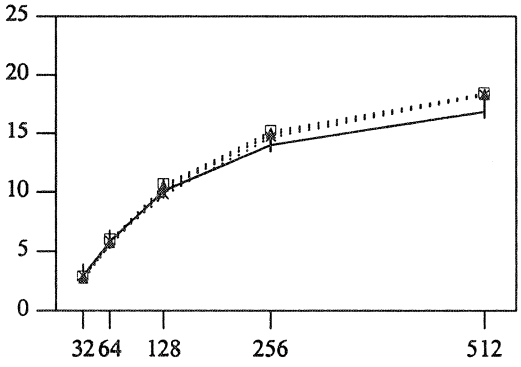

Fig.3c, IBM : Timing data for DGETRI

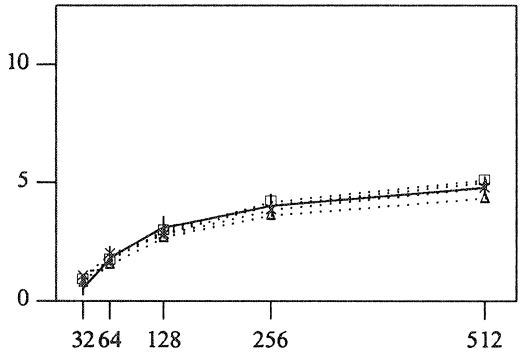

Fig.3d, FX4 : Timing data for ZGETRI

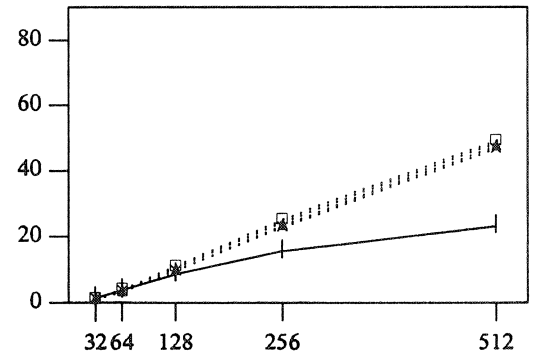

Fig.3e, 205 : Timing data for SGETRI

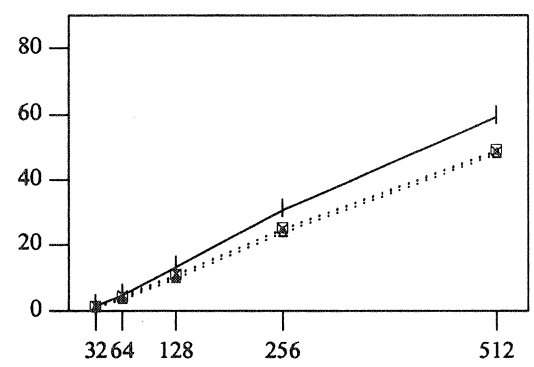

Fig.3f, 205.opt : Timing data for SGETRI

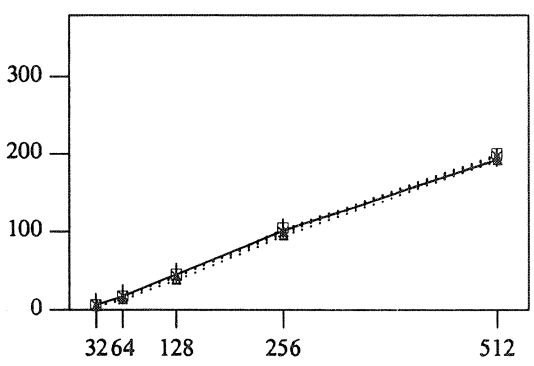

Fig.3g, NEC : Timing data for DGETRI

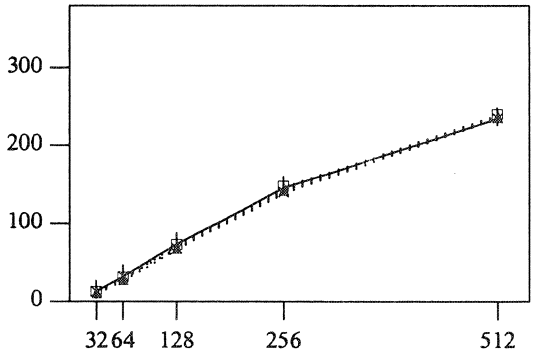

Fig.3h, NEC : Timing data for ZGETRI

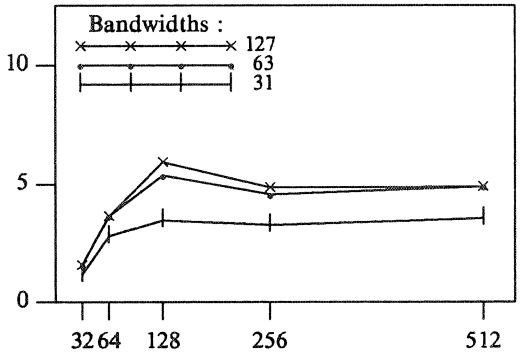

Fig.4a, FX4 : Timing data for DGBTRF

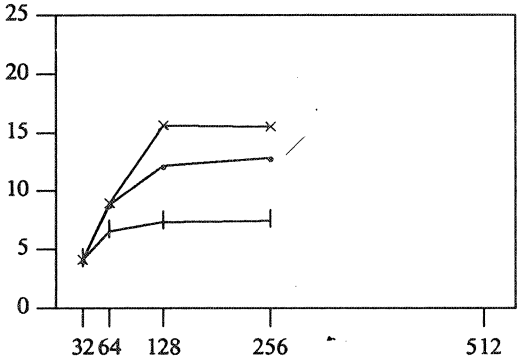

Fig.4b, 995 : Timing data for SGBTRF

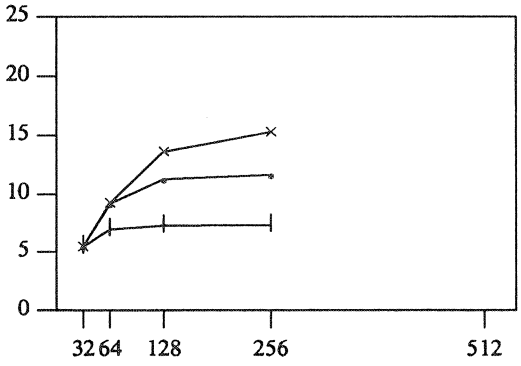

Fig.4c, IBM : Timing data for DGBTRF

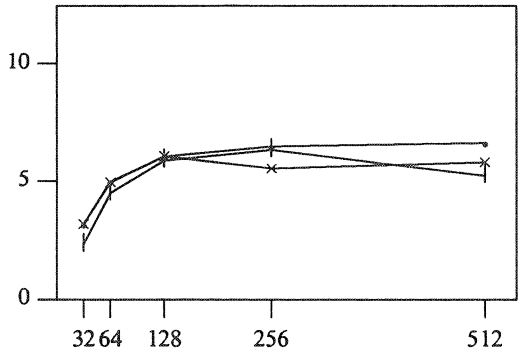

Fig.4d, FX4 : Timing data for ZGBTRF

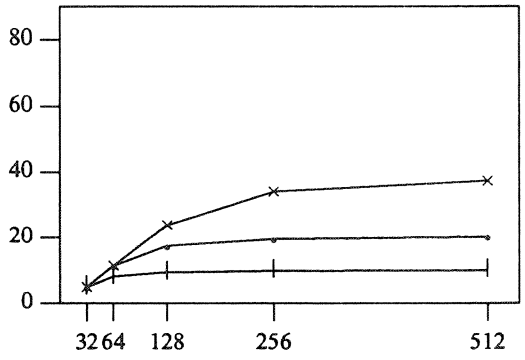

Fig.4e, 205 : Timing data for SGBTRF

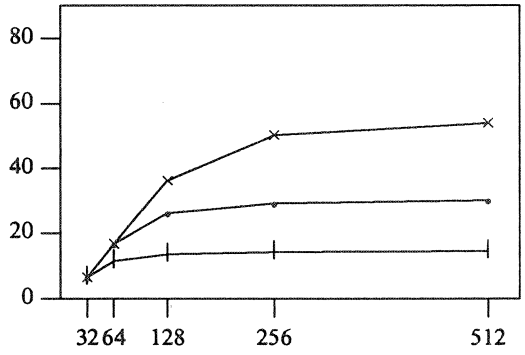

Fig.4f, 205.opt : Timing data for SGBTRF

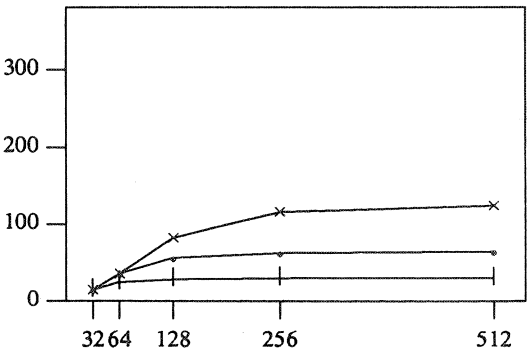

Fig.4g, NEC : Timing data for DGBTRF

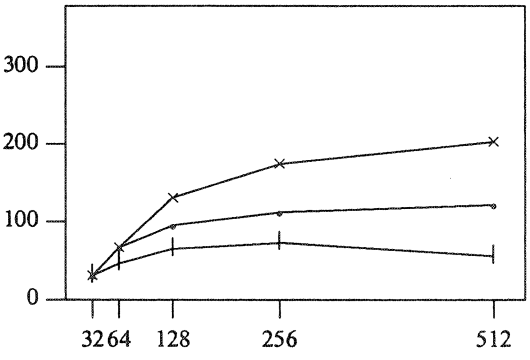

Fig.4h, NEC : Timing data for ZGBTRF

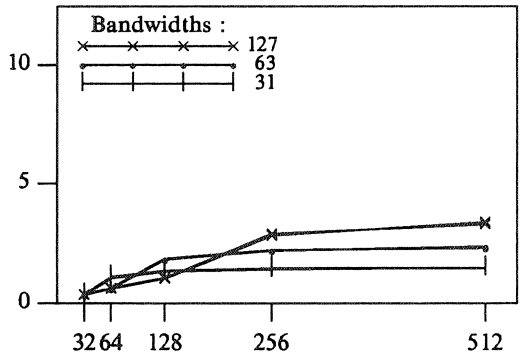

Fig.5a, FX4 : Timing data for DGBTRS

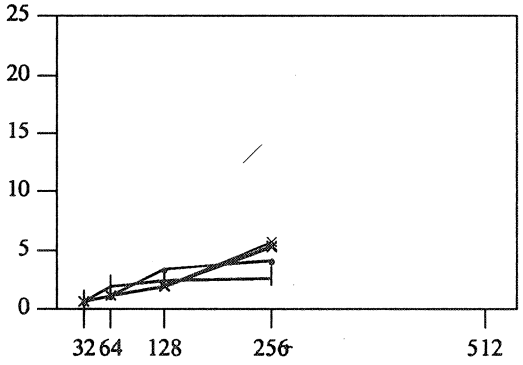

Fig.5b, 995 : Timing data for SGBTRS

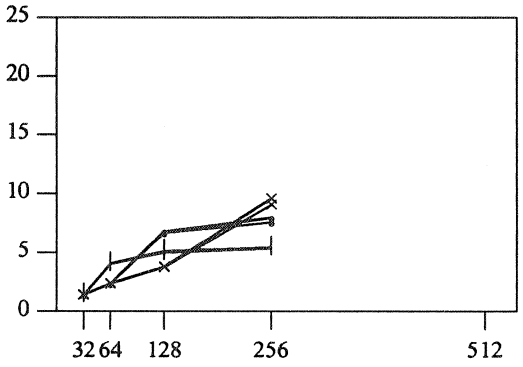

Fig.5c, IBM : Timing data for DGBTRS

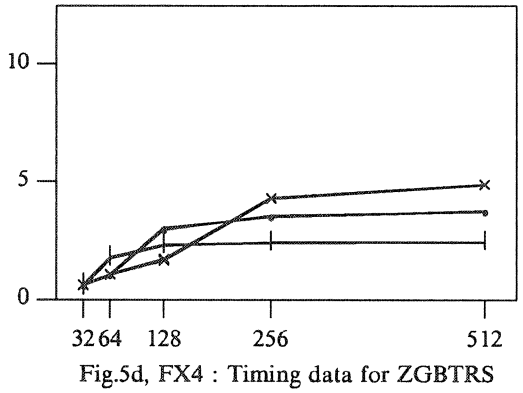

Fig.5d, FX4 : Timing data for ZGBTRS

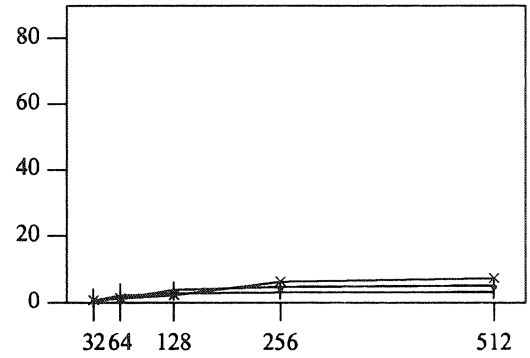

Fig.5e, 205 : Timing data for SGBTRS

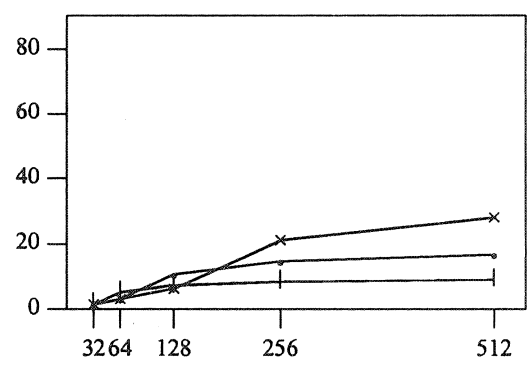

Fig.5f, 205.opt : Timing data for SGBTRS

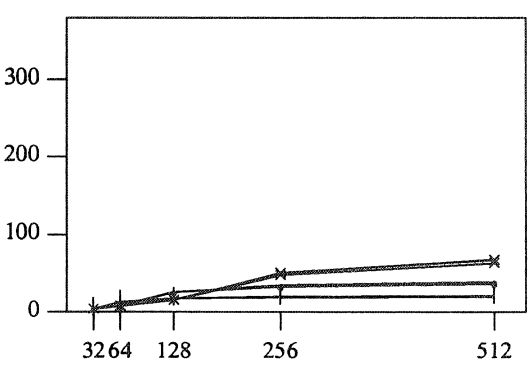

Fig.5g, NEC : Timing data for DGBTRS

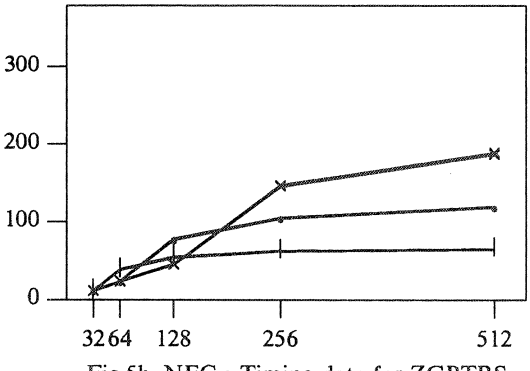

Fig.5h, NEC : Timing data for ZGBTRS

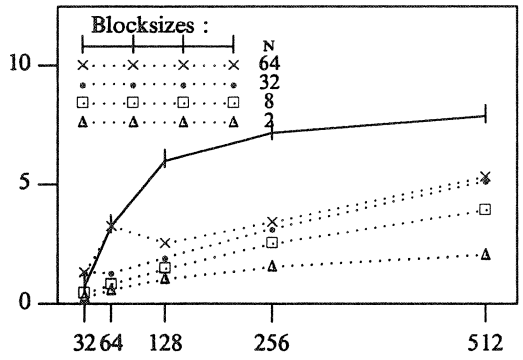

Fig.6a, FX4 : Timing data for DPOTRF, Upper

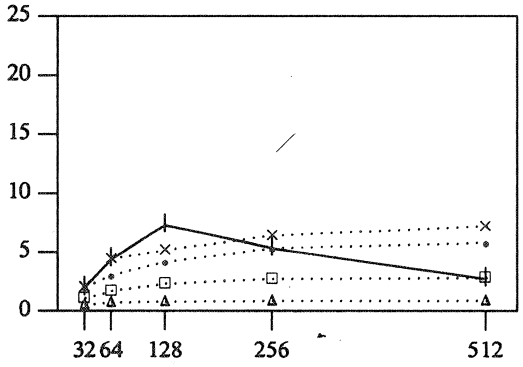

Fig.6b, 995 : Timing data for SPOTRF, Upper

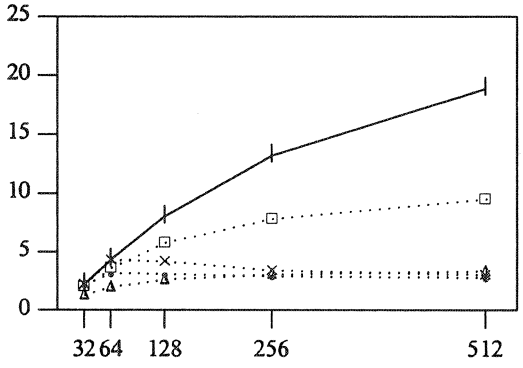

Fig.6c, IBM : Timing data for DPOTRF, Upper

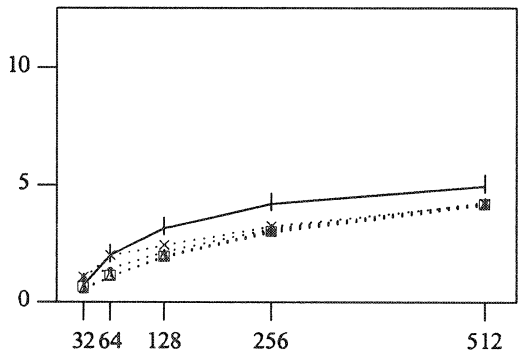

Fig.6d, FX4 : Timing data for ZPOTRF, Upper

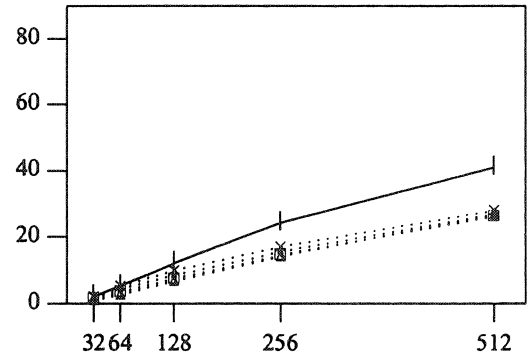

Fig.6e, 205 : Timing data for SPOTRF, Upper

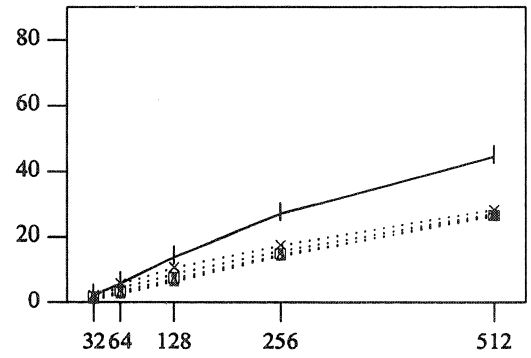

Fig.6f, 205.opt : Timing data for SPOTRF, Upper

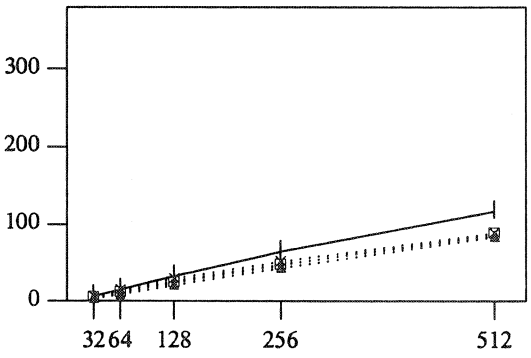

Fig.6g, NEC : Timing data for DPOTRF, Upper

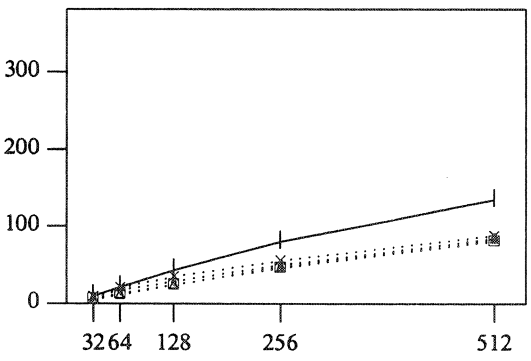

Fig.6h, NEC : Timing data for ZPOTRF, Upper

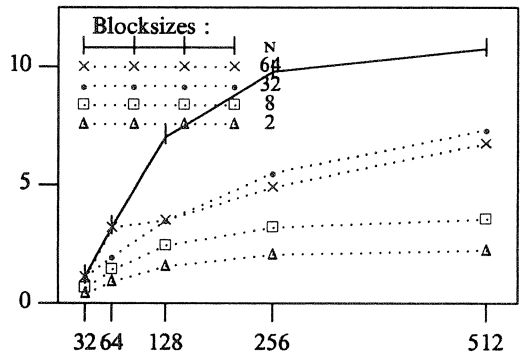

Fig.7a, FX4 : Timing data for DPOTRF, Lower

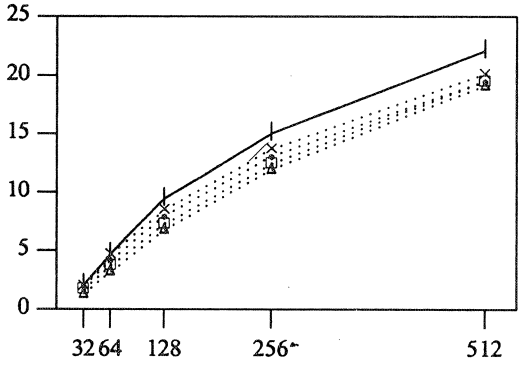

Fig.7b, 995 : Timing data for SPOTRF, Lower

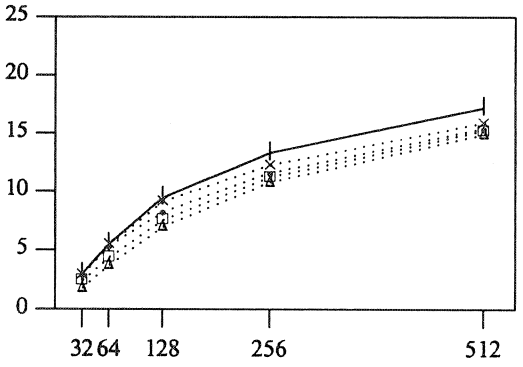

Fig.7c, IBM : Timing data for DPOTRF, Lower

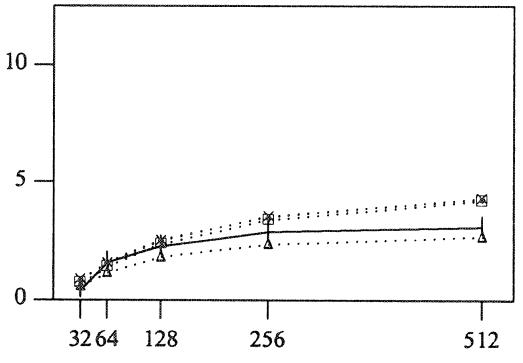

Fig.7d, FX4 : Timing data for ZPOTRF, Lower

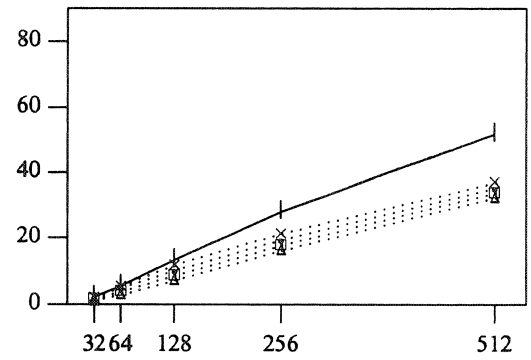

Fig.7e, 205 : Timing data for SPOTRF, Lower

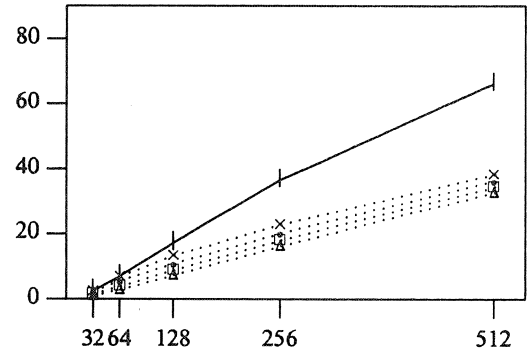

Fig.7f, 205.opt : Timing data for SPOTRF, Lower

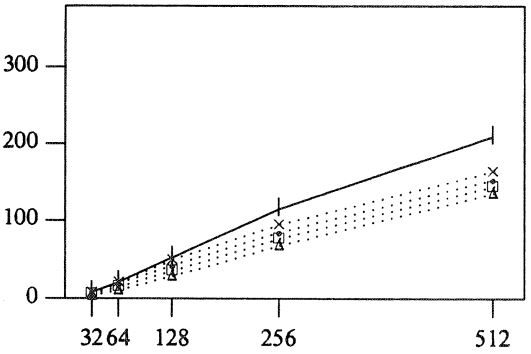

Fig.7g, NEC : Timing data for DPOTRF, Lower

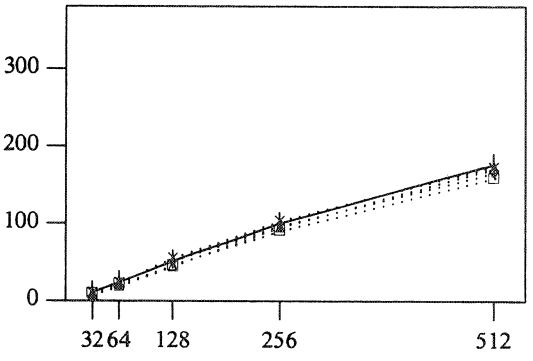

Fig.7h, NEC : Timing data for ZPOTRF, Lower

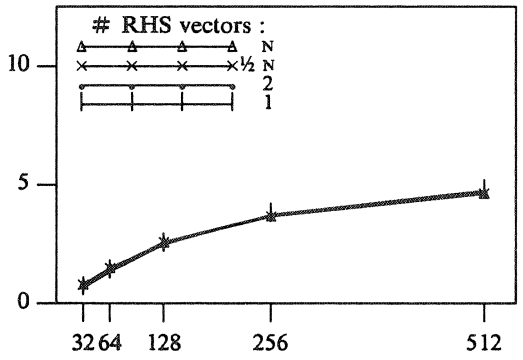

Fig.8a, FX4 : Timing data for DPOTRS, Upper

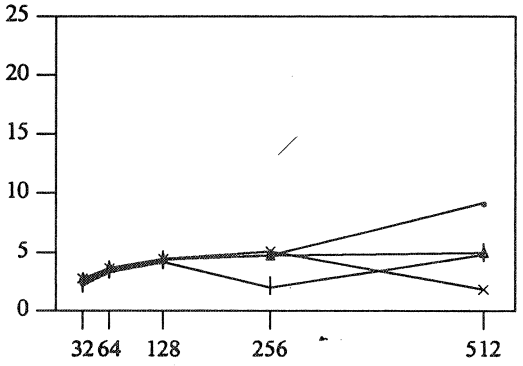

Fig.8b, 995 : Timing data for SPOTRS, Upper

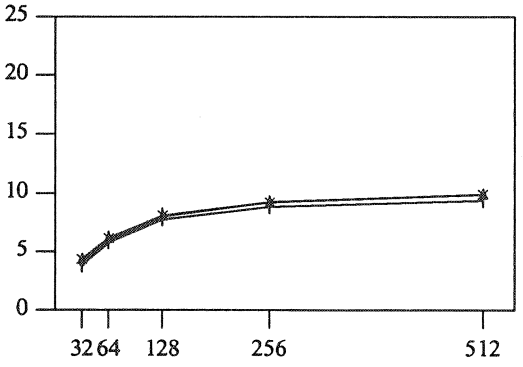

Fig.8c, IBM : Timing data for DPOTRS, Upper

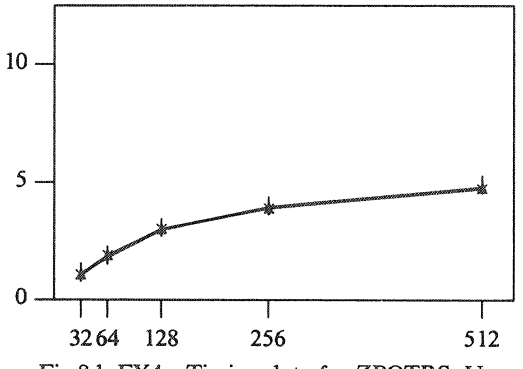

Fig.8d, FX4 : Timing data for ZPOTRS, Upper

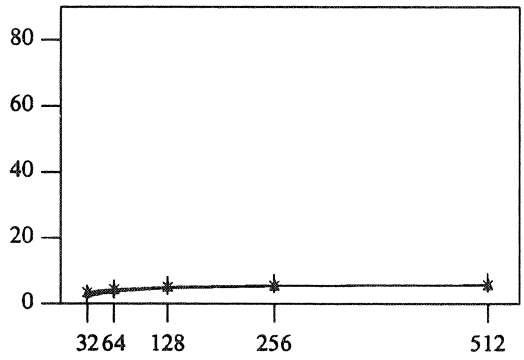

Fig.8e, 205 : Timing data for SPOTRS, Upper

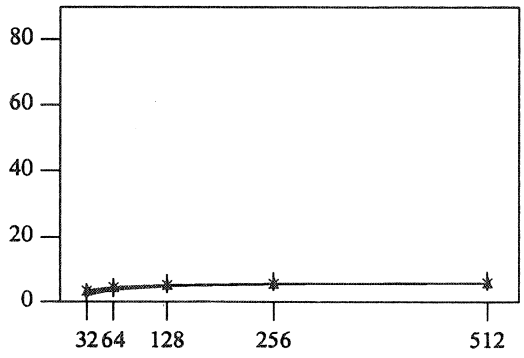

Fig.8f, 205.opt : Timing data for SPOTRS, Upper

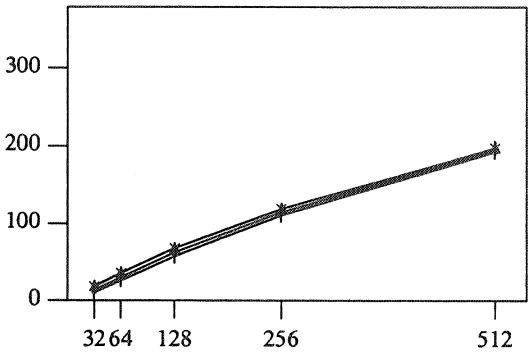

Fig.8g, NEC : Timing data for DPOTRS, Upper

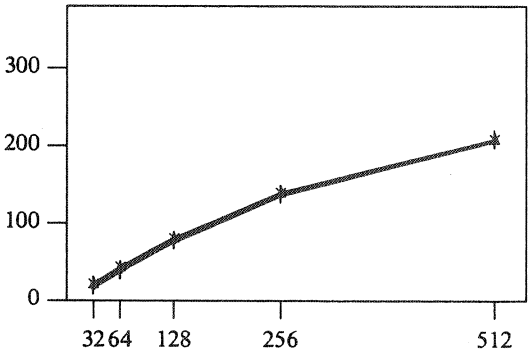

Fig.8h, NEC : Timing data for ZPOTRS, Upper

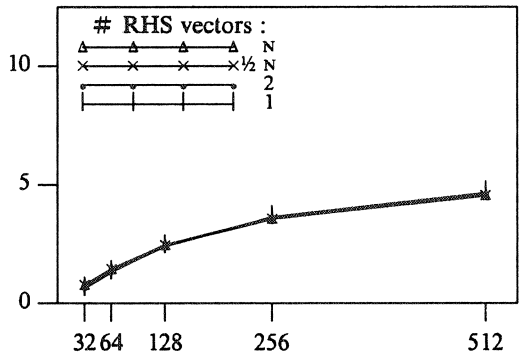

Fig.9a, FX4 : Timing data for DPOTRS, Lower

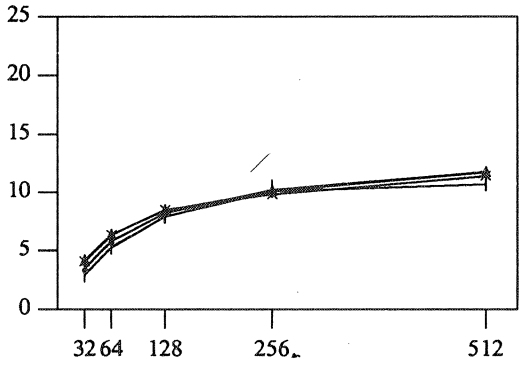

Fig.9b, 995 : Timing data for SPOTRS, Lower

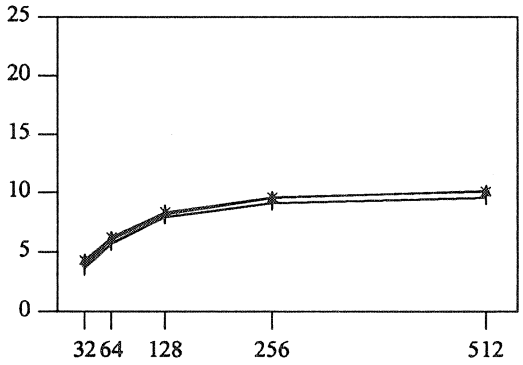

Fig.9c, IBM : Timing data for DPOTRS, Lower

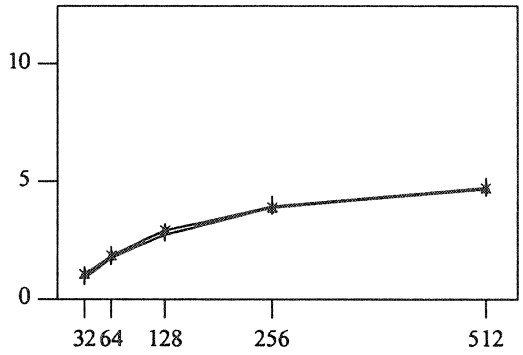

Fig.9d, FX4 : Timing data for ZPOTRS, Lower

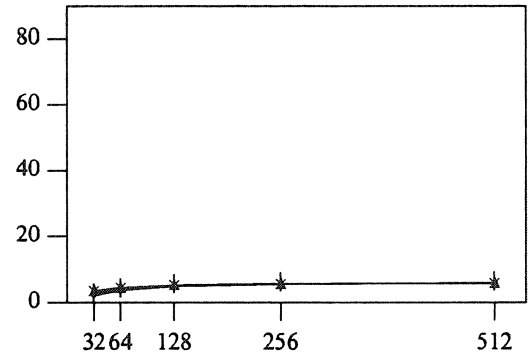

Fig.9e, 205 : Timing data for SPOTRS, Lower

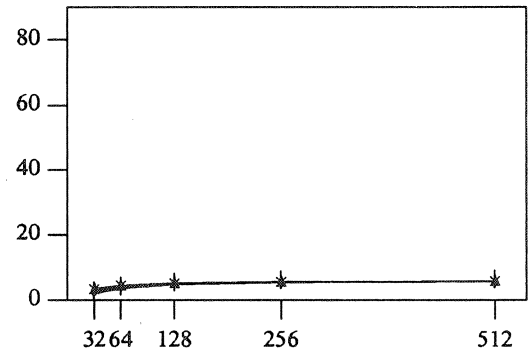

Fig.9f, 205.opt : Timing data for SPOTRS, Lower

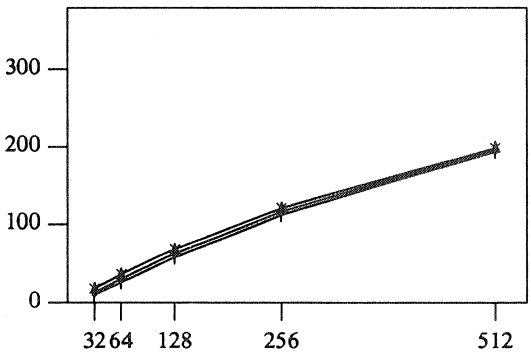

Fig.9g, NEC : Timing data for DPOTRS, Lower

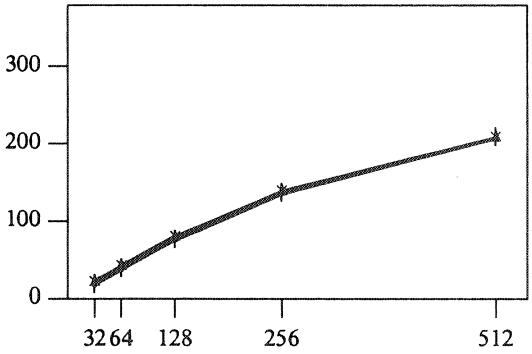

Fig.9h, NEC : Timing data for ZPOTRS, Lower

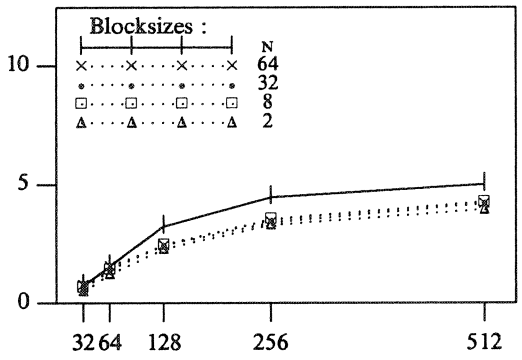

Fig.10a, FX4 : Timing data for DPOTRI, Upper

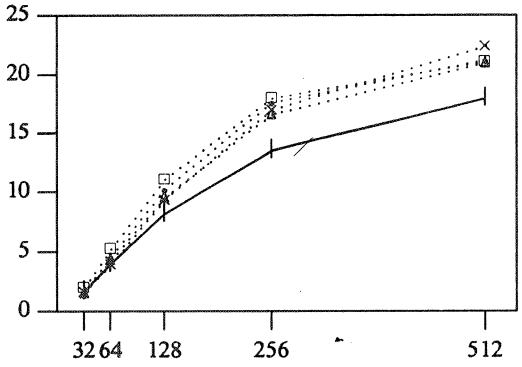

Fig.10b, 995 : Timing data for SPOTRI, Upper

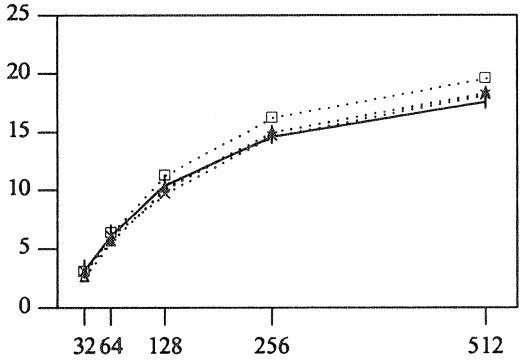

Fig.10c, IBM : Timing data for DPOTRI, Upper

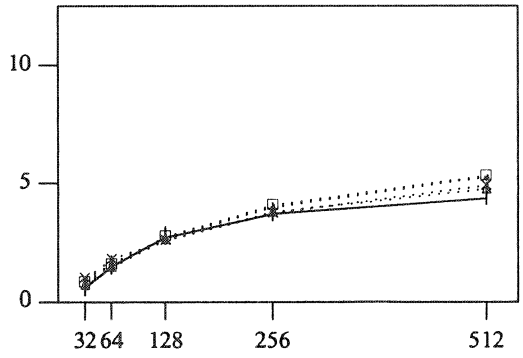

Fig.10d, FX4 : Timing data for ZPOTRI, Upper

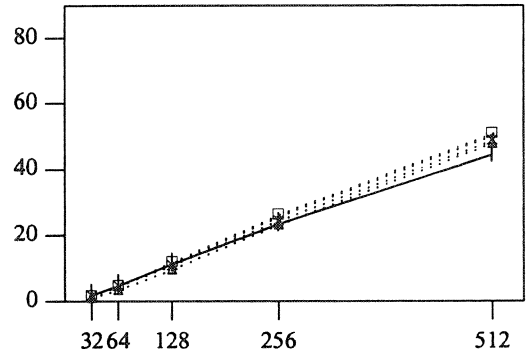

Fig.10e, 205 : Timing data for SPOTRI, Upper

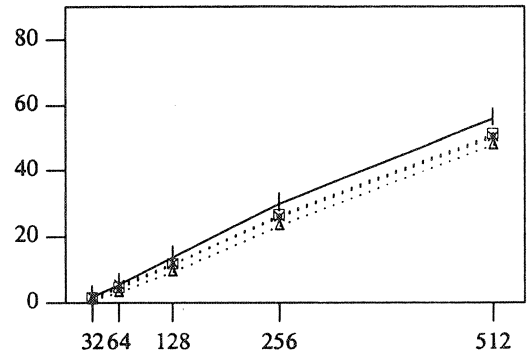

Fig.10f, 205.opt : Timing data for SPOTRI, Upper

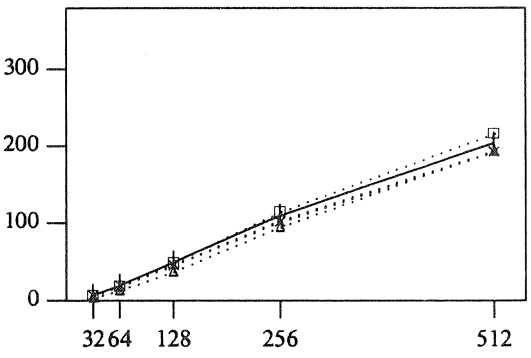

Fig.10g, NEC : Timing data for DPOTRI, Upper

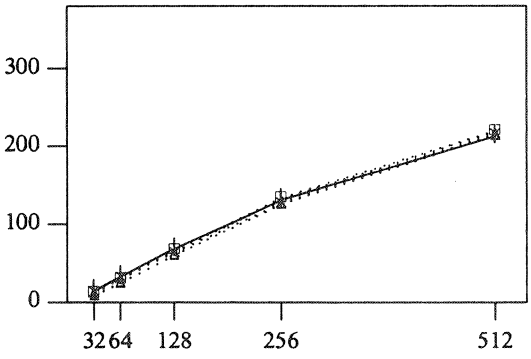

Fig.10h, NEC : Timing data for ZPOTRI, Upper

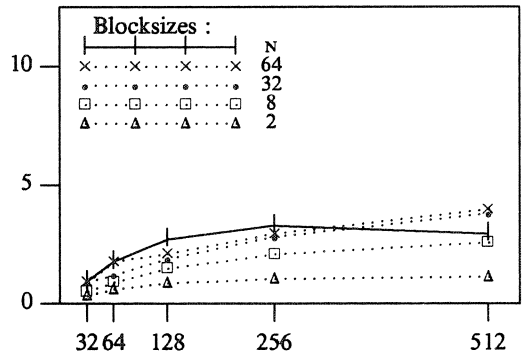

Fig.11a, FX4 : Timing data for DPOTRI, Lower

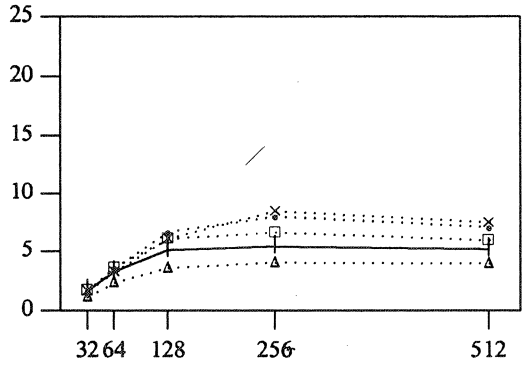

Fig.11b, 995 : Timing data for SPOTRI, Lower

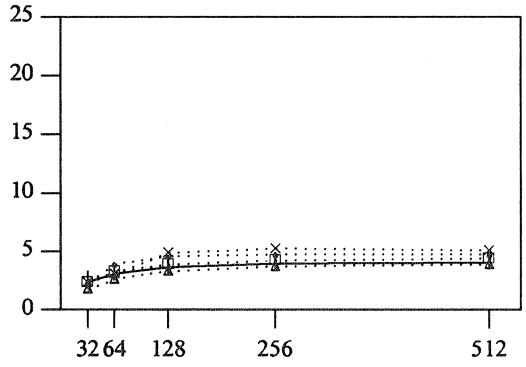

Fig.11c, IBM : Timing data for DPOTRI, Lower

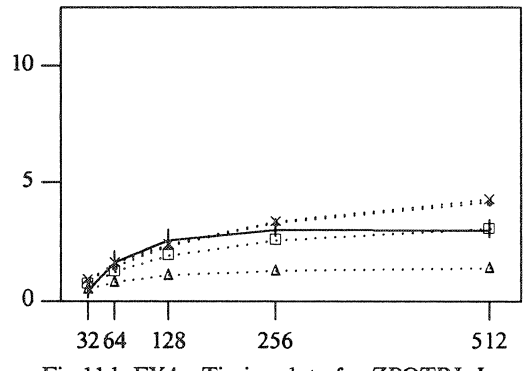

Fig.11d, FX4 : Timing data for ZPOTRI, Lower

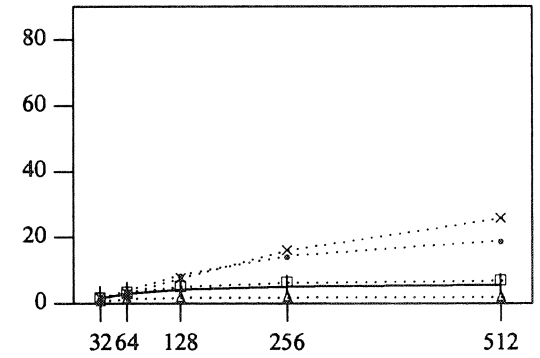

Fig.11e, 205 : Timing data for SPOTRI, Lower

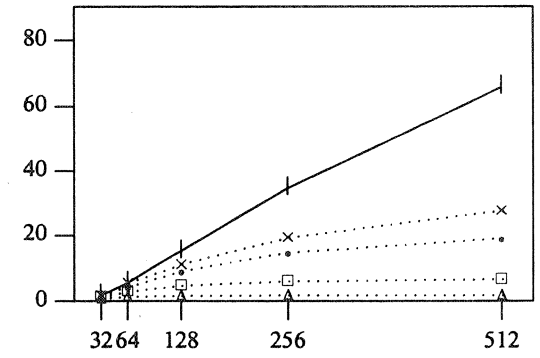

Fig.11f, 205.opt : Timing data for SPOTRI, Lower

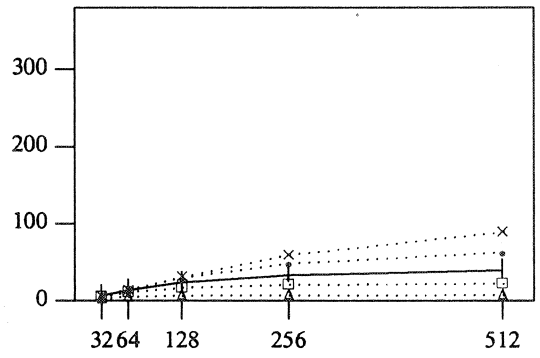

Fig.11g, NEC : Timing data for DPOTRI, Lower

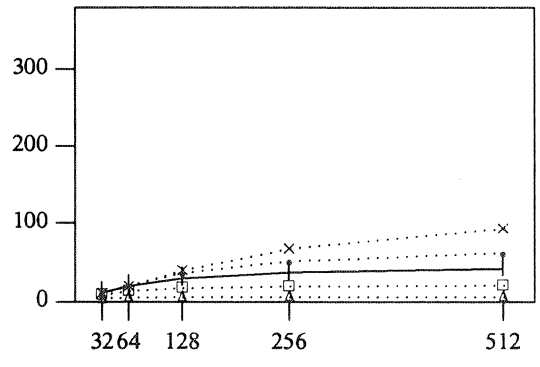

Fig.11h, NEC : Timing data for ZPOTRI, Lower

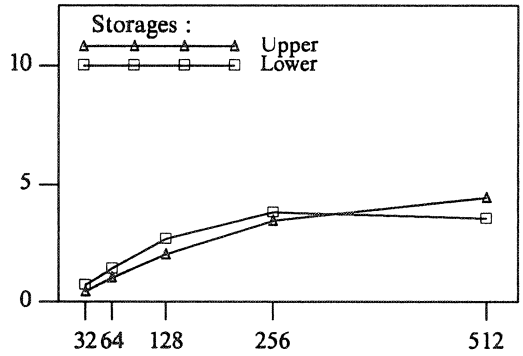

Fig.12a, FX4 : Timing data for DPPTRF

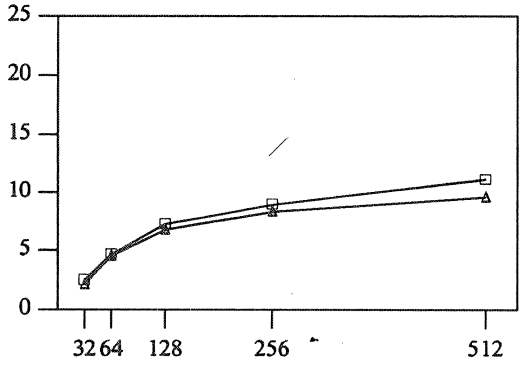

Fig.12b, 995 : Timing data for SPPTRF

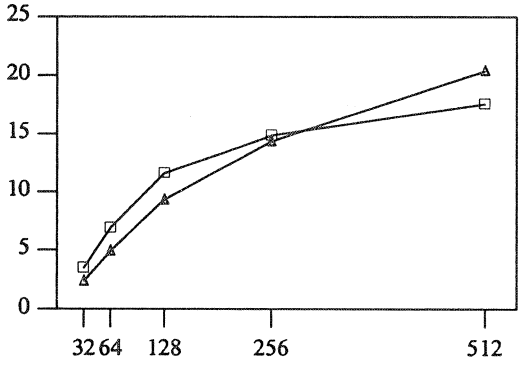

Fig.12c, IBM : Timing data for DPPTRF

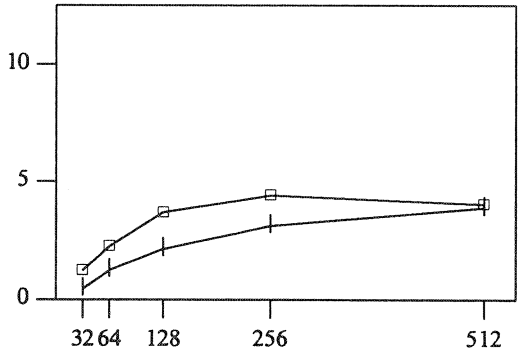

Fig.12d, FX4 : Timing data for ZPPTRF

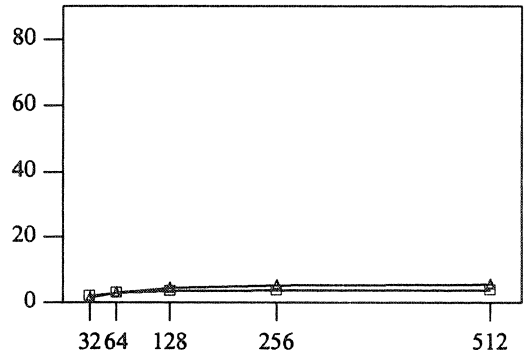

Fig.12e, 205 : Timing data for SPPTRF

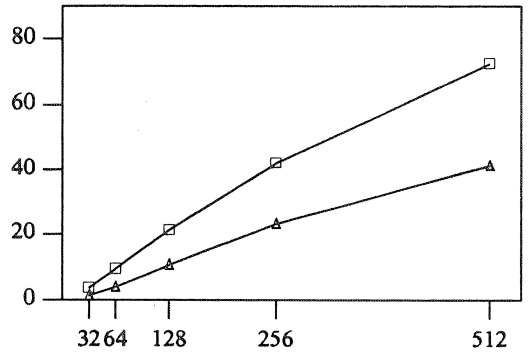

Fig.12f, 205.opt : Timing data for SPPTRF

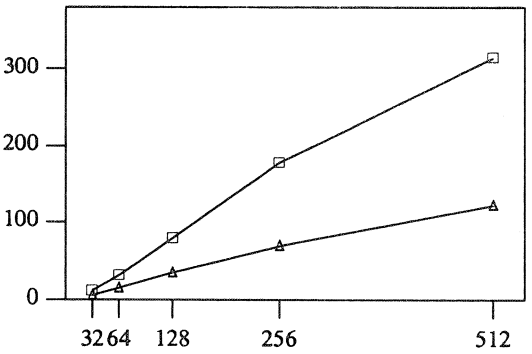

Fig.12g, NEC : Timing data for DPPTRF

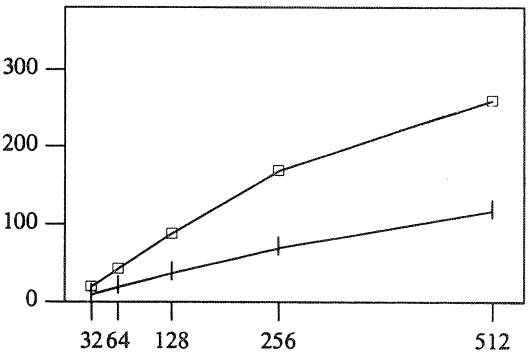

Fig.12h, NEC : Timing data for ZPPTRF

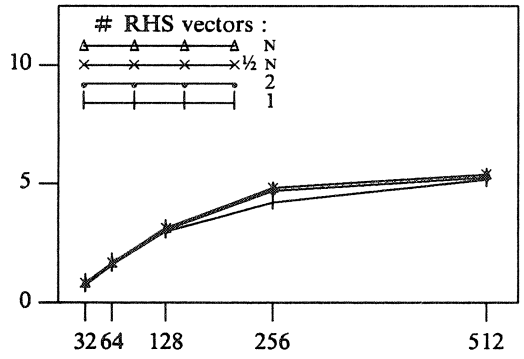

Fig.13a, FX4 : Timing data for DPPTRS, Upper

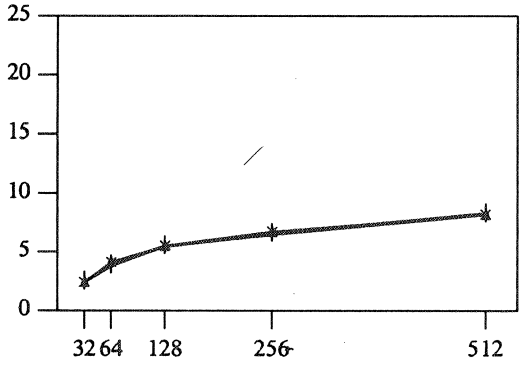

Fig.13b, 995 : Timing data for SPPTRS, Upper

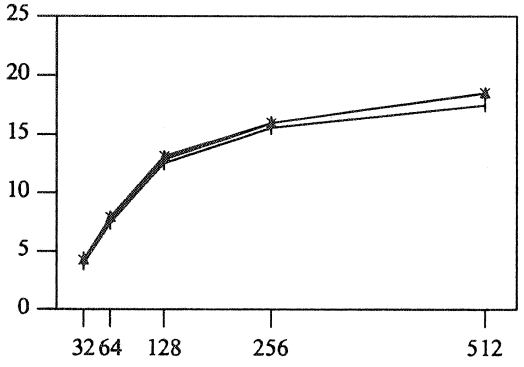

Fig.13c, IBM : Timing data for DPPTRS, Upper

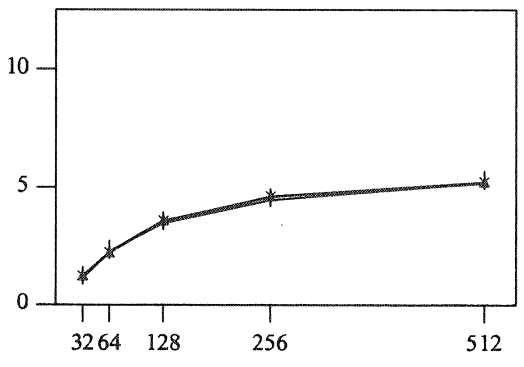

Fig.13d, FX4 : Timing data for ZPPTRS, Upper

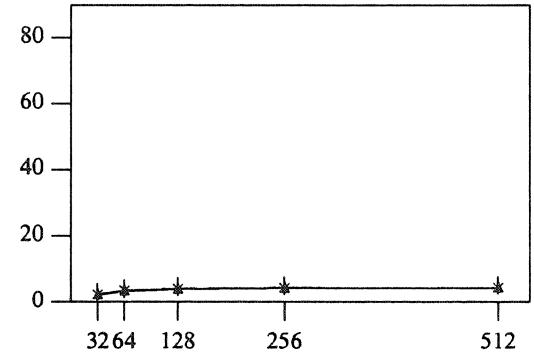

Fig.13e, 205 : Timing data for SPPTRS, Upper

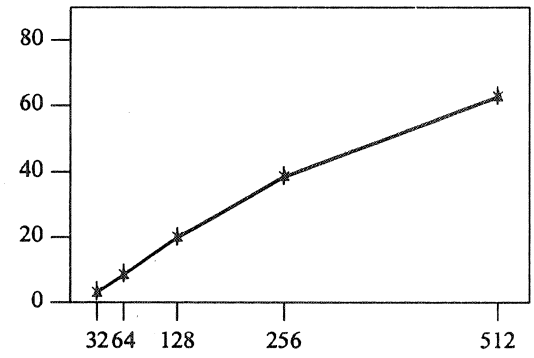

Fig.13f, 205.opt : Timing data for SPPTRS, Upper

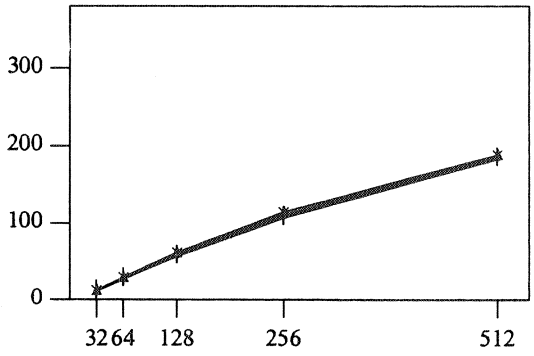

Fig.13g, NEC : Timing data for DPPTRS, Upper

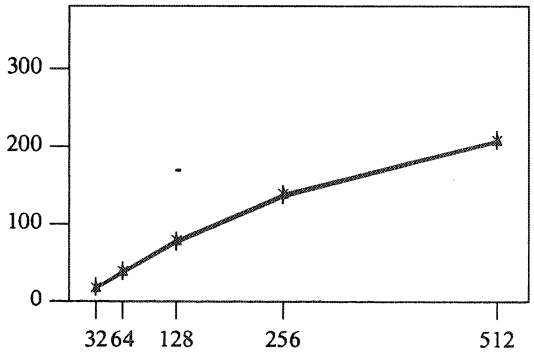

Fig.13h, NEC : Timing data for ZPPTRS, Upper

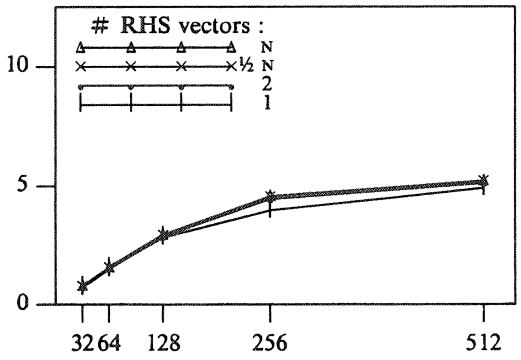

Fig.14a, FX4 : Timing data for DPPTRS, Lower

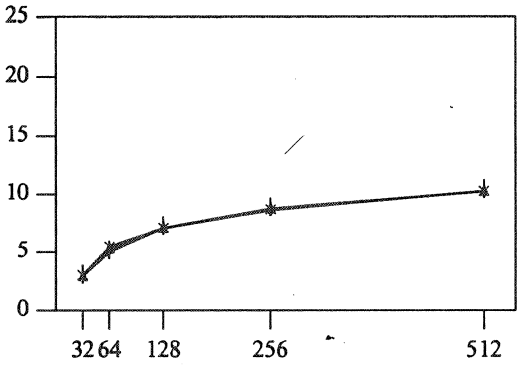

Fig.14b, 995 : Timing data for SPPTRS, Lower

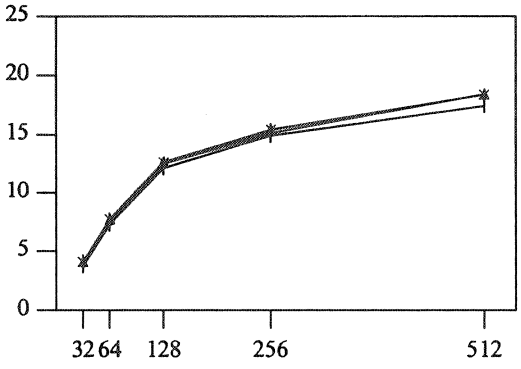

Fig.14c, IBM : Timing data for DPPTRS, Lower

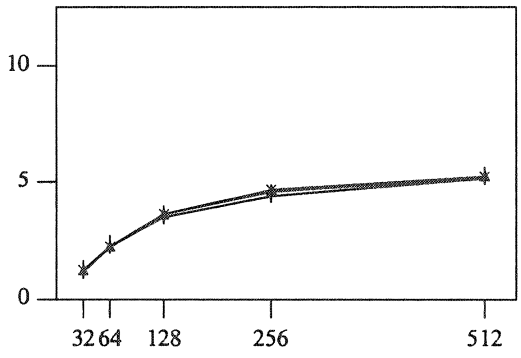

Fig.14d, FX4 : Timing data for ZPPTRS, Lower

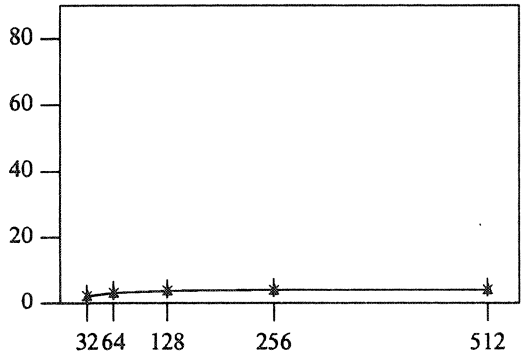

Fig.14e, 205 : Timing data for SPPTRS, Lower

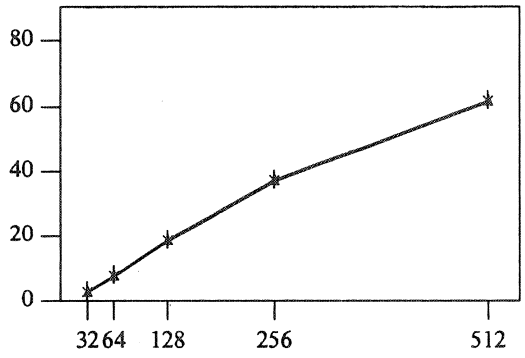

Fig.14f, 205.opt : Timing data for SPPTRS, Lower

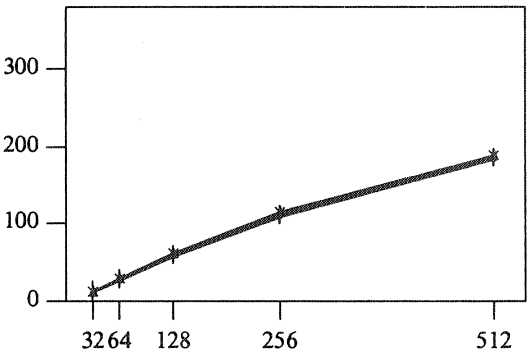

Fig.14g, NEC : Timing data for DPPTRS, Lower

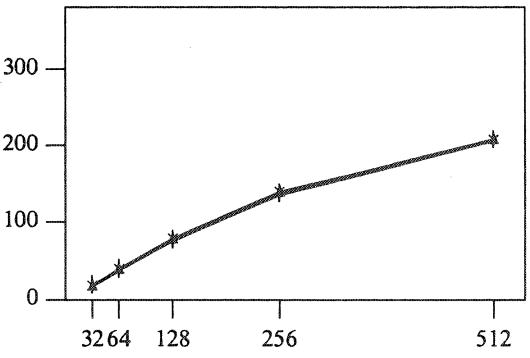

Fig.14h, NEC : Timing data for ZPPTRS, Lower

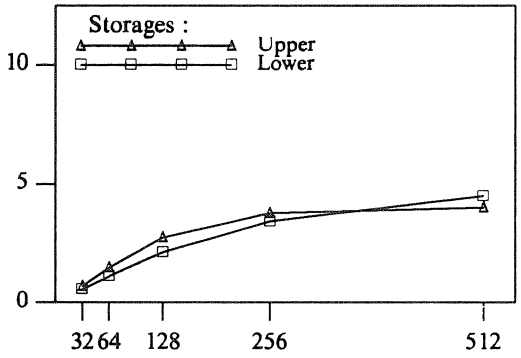

Fig.15a, FX4 : Timing data for DPPTRI

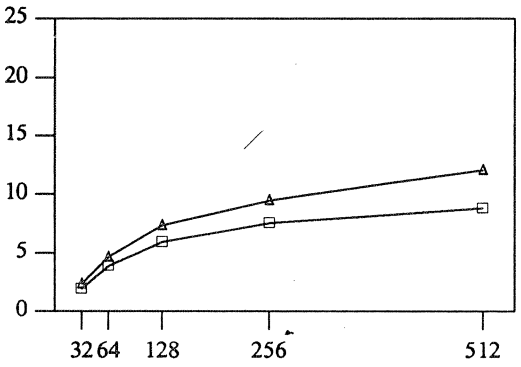

Fig.15b, 995 : Timing data for SPPTRI

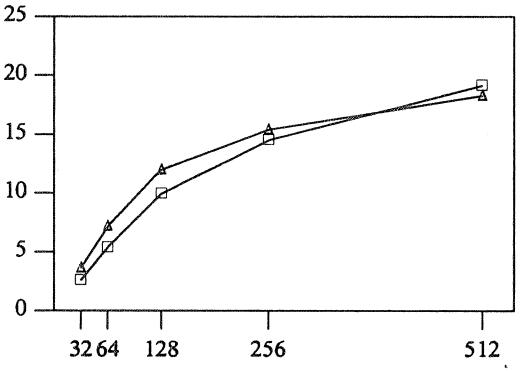

Fig.15c, IBM : Timing data for DPPTRI

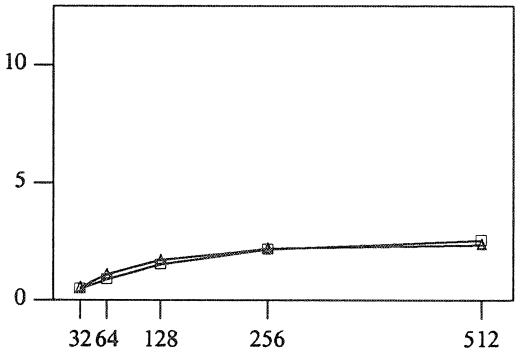

Fig.15d, FX4 : Timing data for ZPPTRI

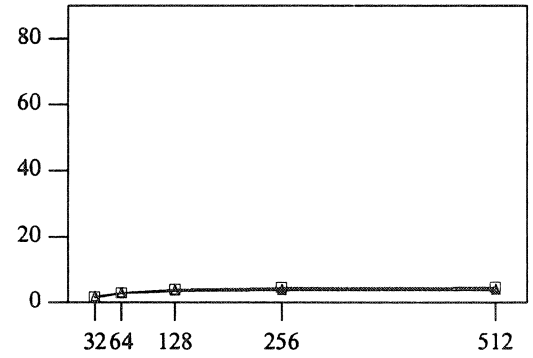

Fig.15e, 205 : Timing data for SPPTRI

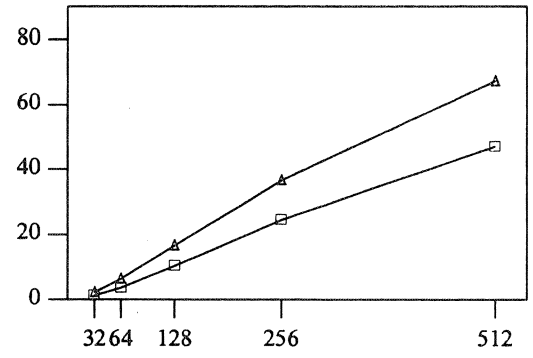

Fig.15f, 205.opt : Timing data for SPPTRI

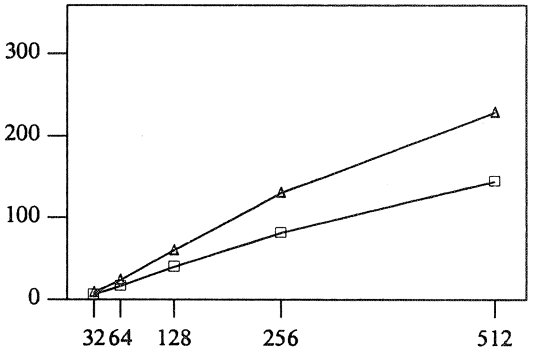

Fig.15g, NEC : Timing data for DPPTRI

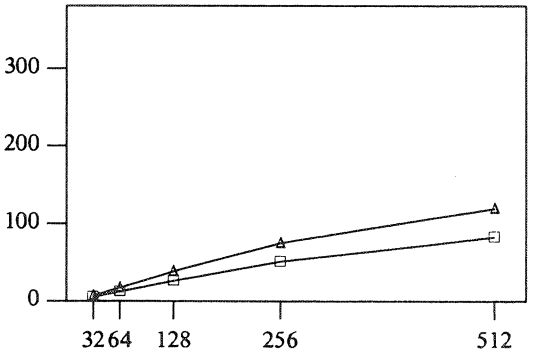

Fig.15h, NEC : Timing data for ZPPTRI

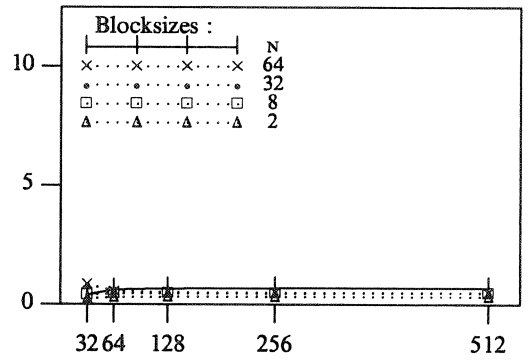

Fig.16a, FX4 : Timing data for DPBTRF, Upper, K=31

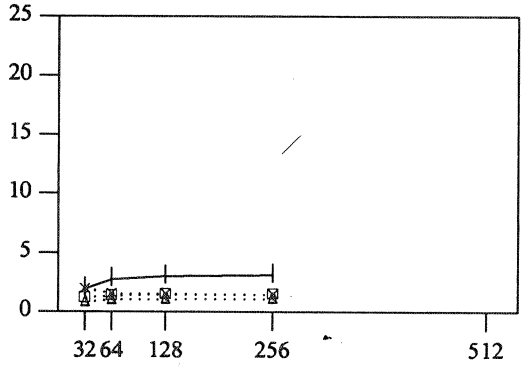

Fig.16b, 995 : Timing data for SPBTRF, Upper, K=31

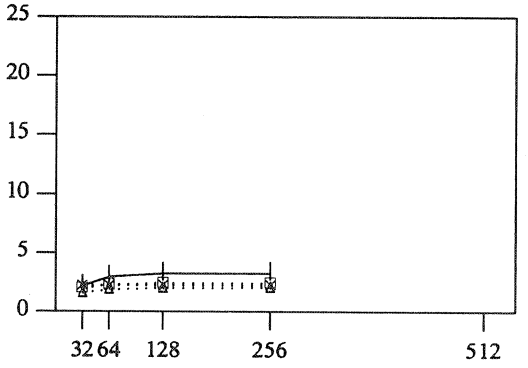

Fig.16c, IBM : Timing data for DPBTRF, Upper, K=31

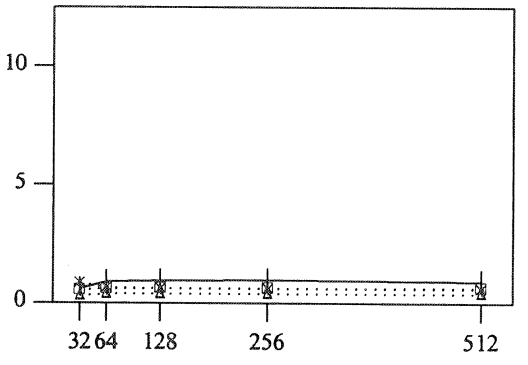

Fig.16d, FX4 : Timing data for ZPBTRF, Upper, K=31

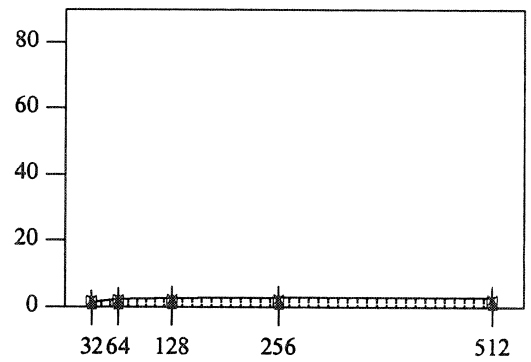

Fig.16e, 205 : Timing data for SPBTRF, Upper, K=31

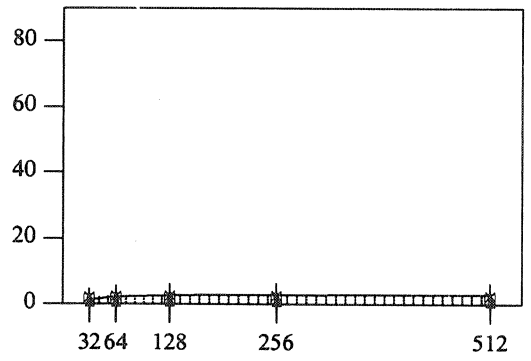

Fig.16f, 205.opt : Timing data for SPBTRF, Upper, K=31

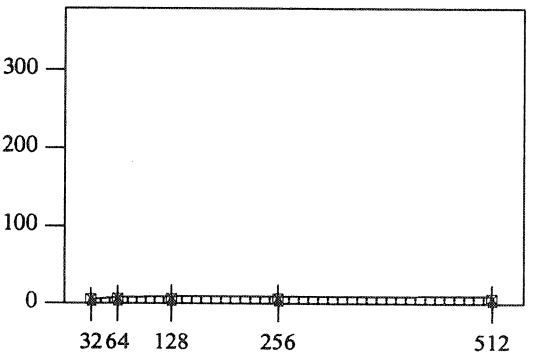

Fig.16g, NEC : Timing data for DPBTRF, Upper, K=31

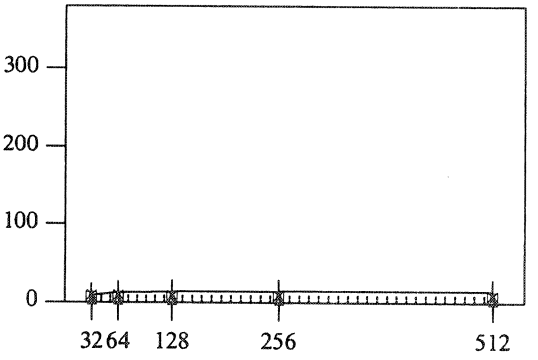

Fig.16h, NEC : Timing data for ZPBTRF, Upper, K=31

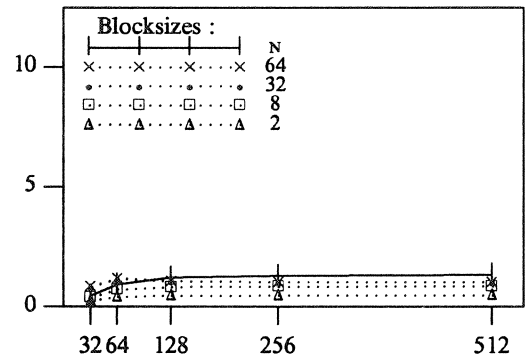

Fig.17a, FX4 : Timing data for DPBTRF, Upper, K=63

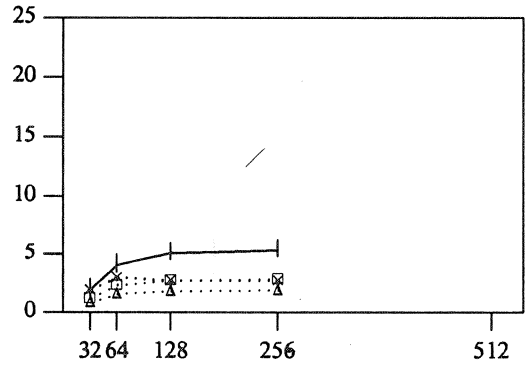

Fig.17b, 995 : Timing data for SPBTRF, Upper, K=63

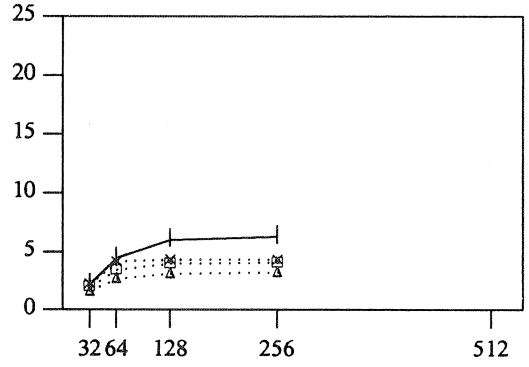

Fig.17c, IBM : Timing data for DPBTRF, Upper, K=63

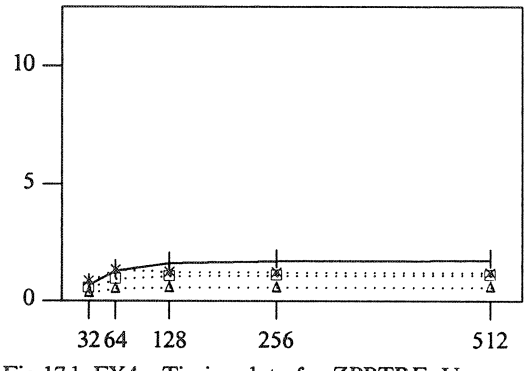

Fig.17d, FX4 : Timing data for ZPBTRF, Upper, K=63

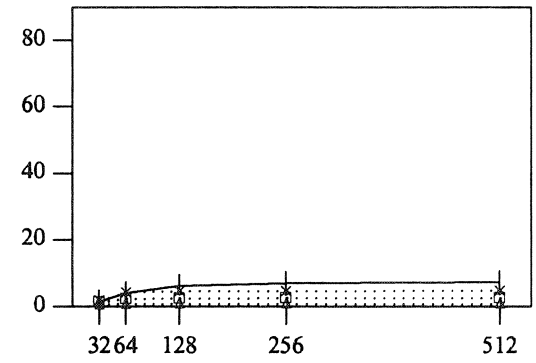

Fig.17e, 205 : Timing data for SPBTRF, Upper, K=63

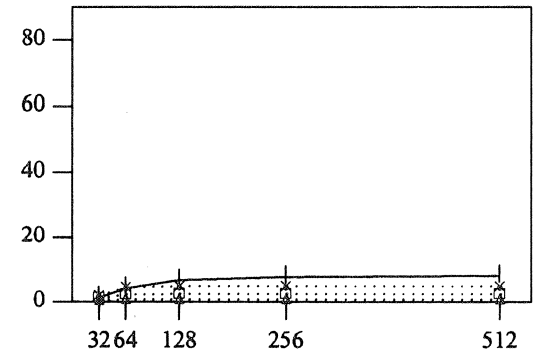

Fig.17f, 205.opt : Timing data for SPBTRF, Upper, K=63

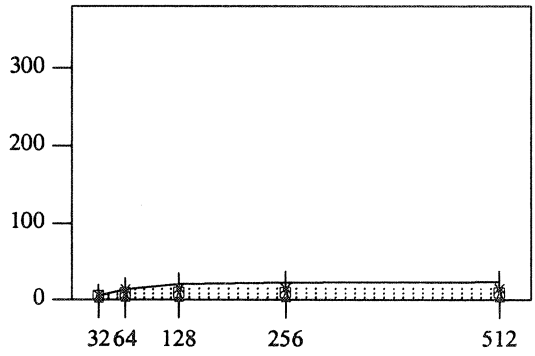

Fig.17g, NEC : Timing data for DPBTRF, Upper, K=63

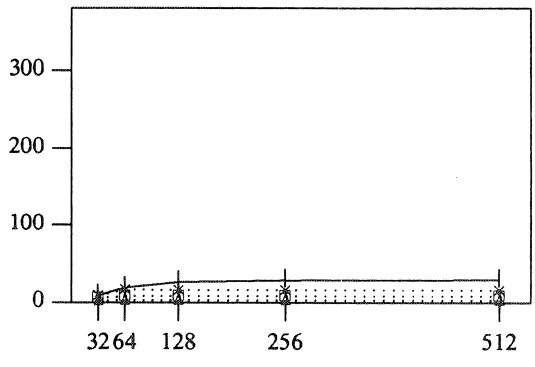

Fig.17h, NEC : Timing data for ZPBTRF, Upper, K=63

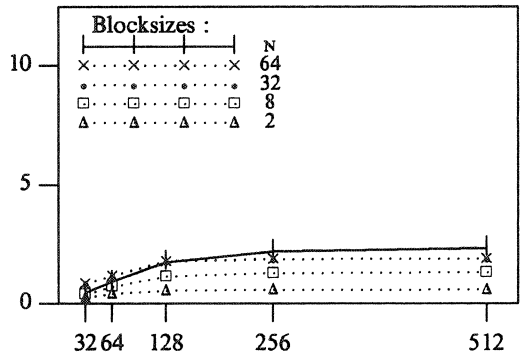

Fig.18a, FX4 : Timing data for DPBTRF, Upper, K = 127

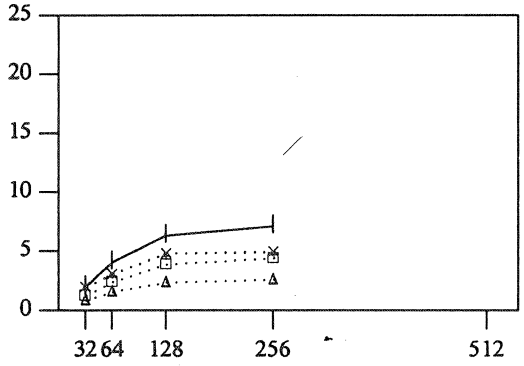

Fig.18b, 995 : Timing data for SPBTRF, Upper, K = 127

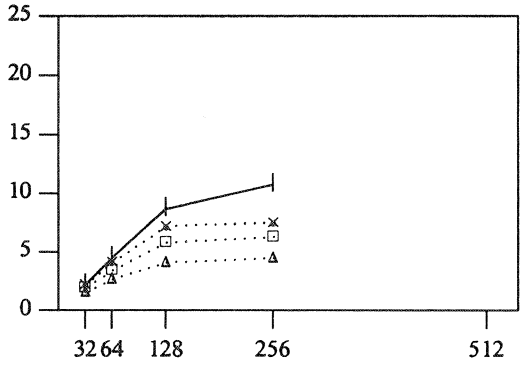

Fig.18c, IBM : Timing data for DPBTRF, Upper, K = 127

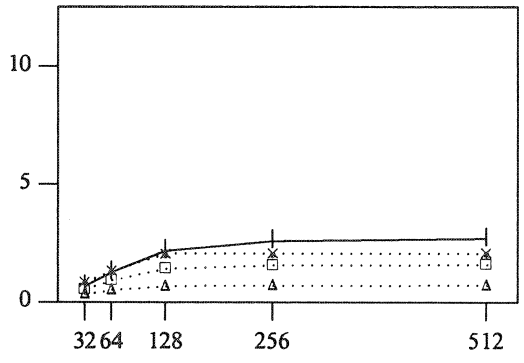

Fig.18d, FX4 : Timing data for ZPBTRF, Upper, K = 127

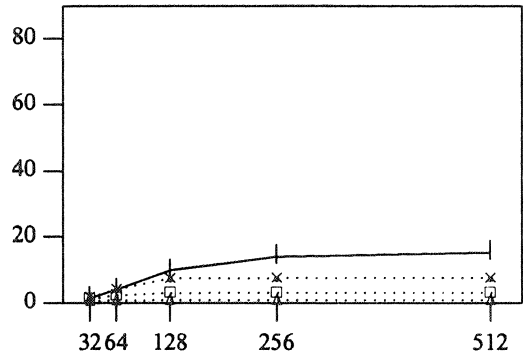

Fig.18e, 205 : Timing data for SPBTRF, Upper, K = 127

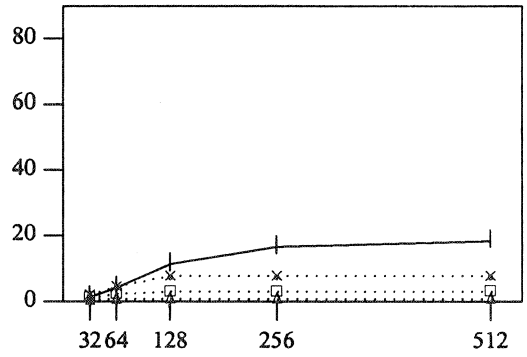

Fig.18f, 205.opt : Timing data for SPBTRF, Upper, K = 127

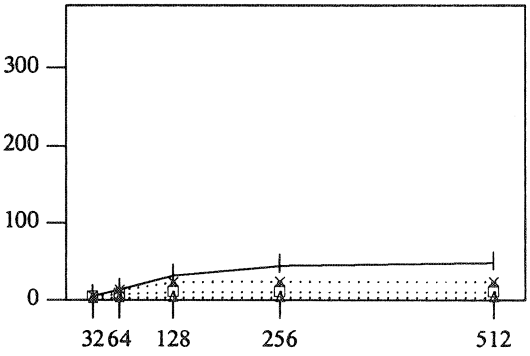

Fig.18g, NEC : Timing data for DPBTRF, Upper, K = 127

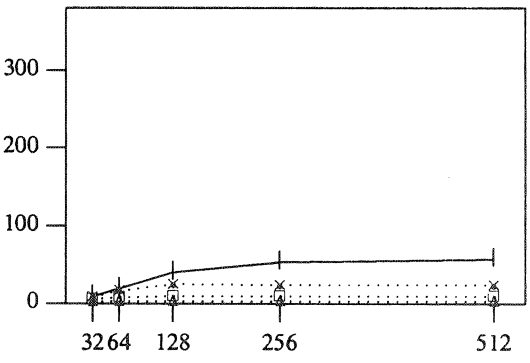

Fig.18h, NEC : Timing data for ZPBTRF, Upper, K = 127

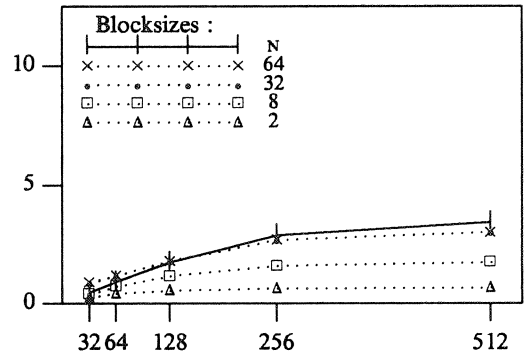

Fig.19a, FX4 : Timing data for DPBTRF, Upper, K=255

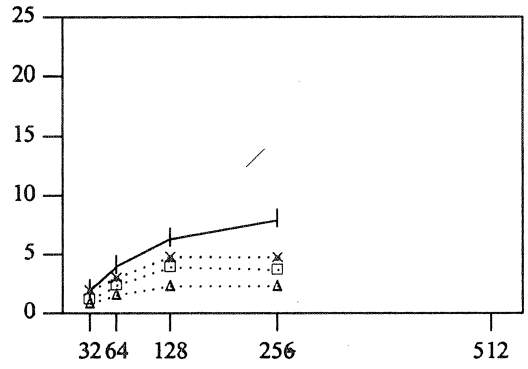

Fig.19b, 995 : Timing data for SPBTRF, Upper, K=255

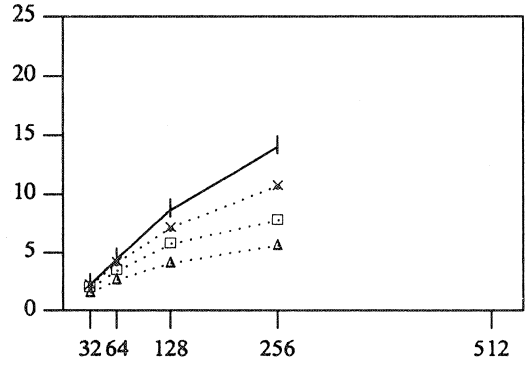

Fig.19c, IBM : Timing data for DPBTRF, Upper, K=255

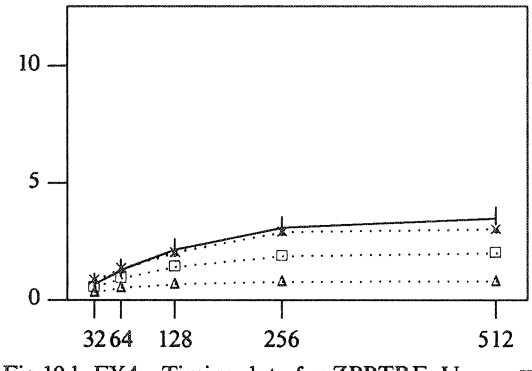

Fig.19d, FX4 : Timing data for ZPBTRF, Upper, K=255

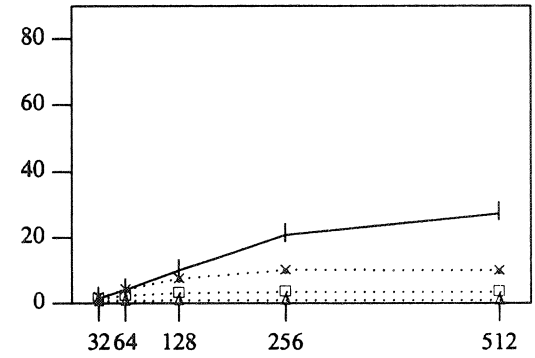

Fig.19e, 205 : Timing data for SPBTRF, Upper, K=255

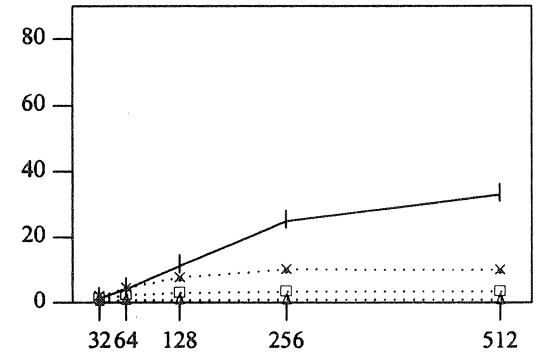

Fig.19f, 205.opt : Timing data for SPBTRF, Upper, K=255

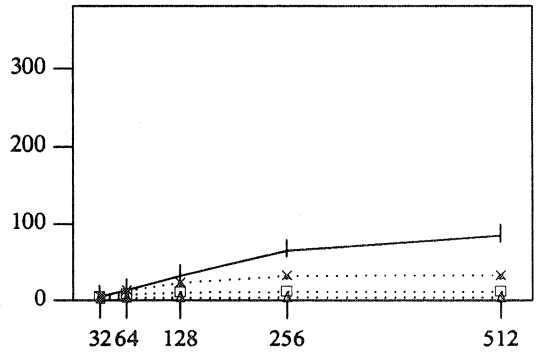

Fig.19g, NEC : Timing data for DPBTRF, Upper, K=255

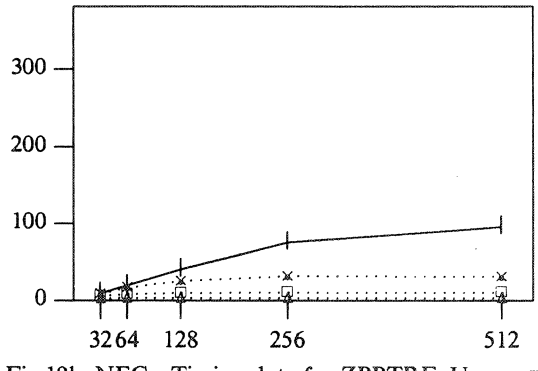

Fig.19h, NEC : Timing data for ZPBTRF, Upper, K=255

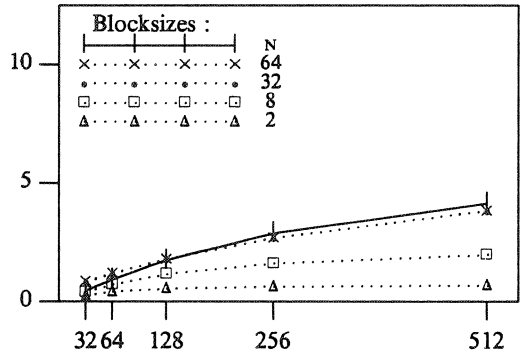

Fig.20a, FX4 : Timing data for DPBTRF, Upper, κ=511

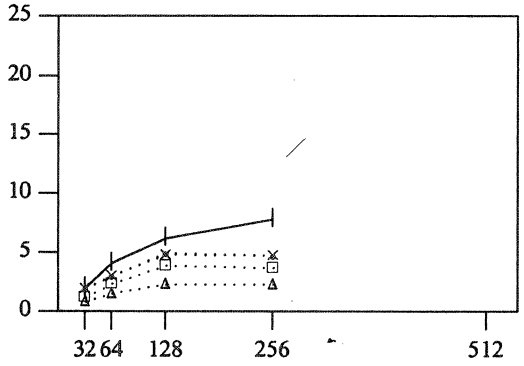

Fig.20b, 995 : Timing data for SPBTRF, Upper, κ=511

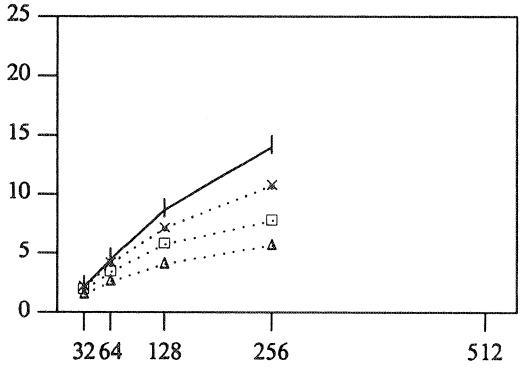

Fig.20c, IBM : Timing data for DPBTRF, Upper, κ=511

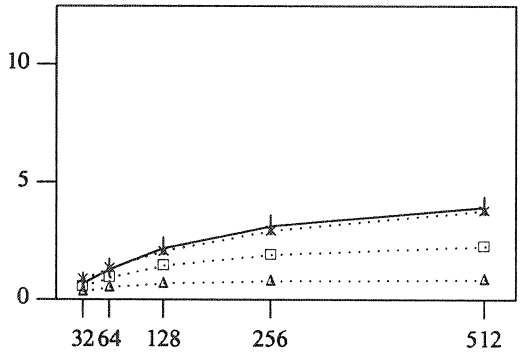

Fig.20d, FX4 : Timing data for ZPBTRF, Upper, κ=511

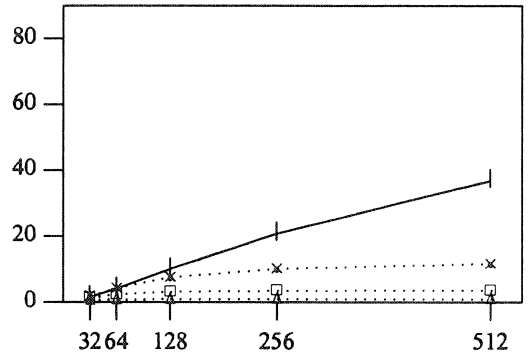

Fig.20e, 205 : Timing data for SPBTRF, Upper, κ=511

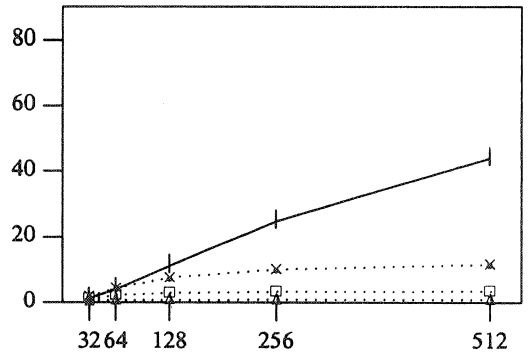

Fig.20f, 205.opt : Timing data for SPBTRF, Upper, κ=511

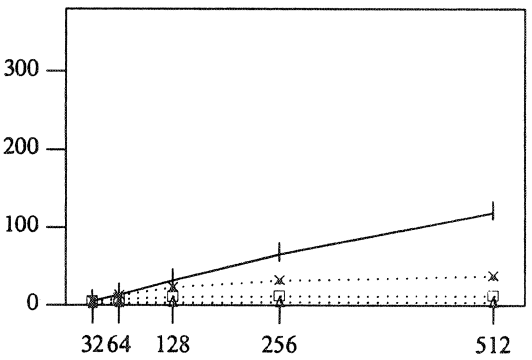

Fig.20g, NEC : Timing data for DPBTRF, Upper, κ=511

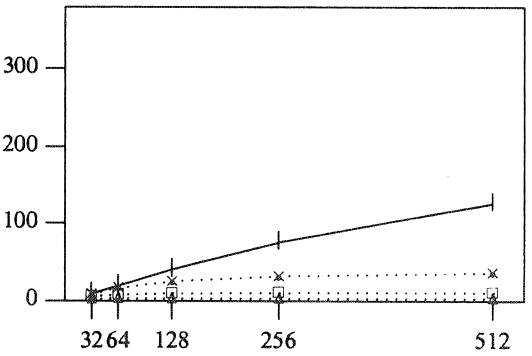

Fig.20h, NEC : Timing data for ZPBTRF, Upper, κ=511

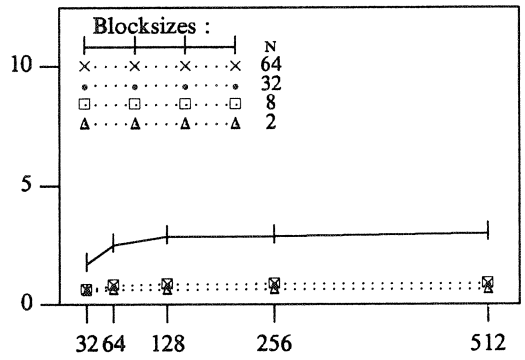

Fig.21a, FX4 : Timing data for DPBTRF, Lower, K = 31

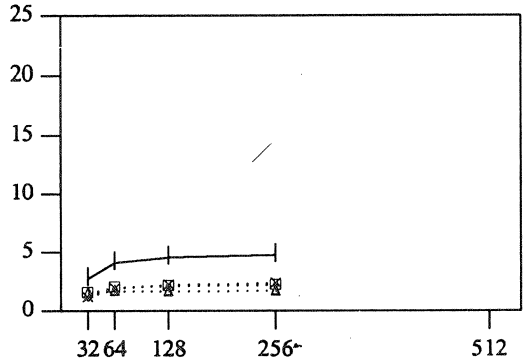

Fig.21b, 995 : Timing data for SPBTRF, Lower, K = 31

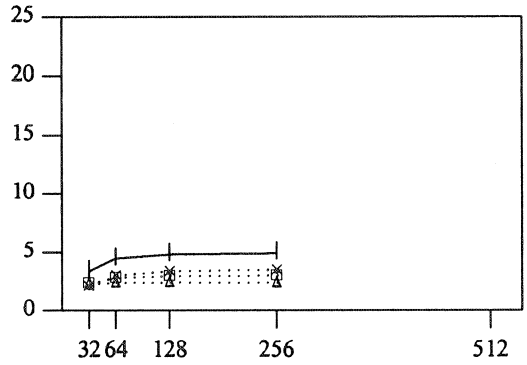

Fig.21c, IBM : Timing data for DPBTRF, Lower, K = 31

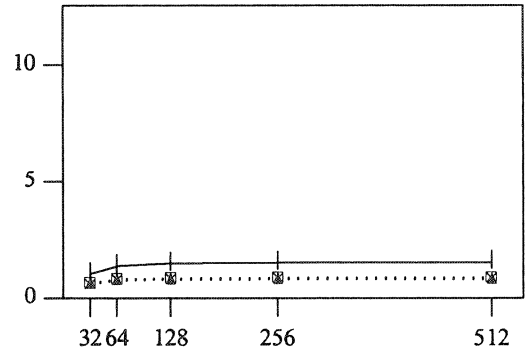

Fig.21d, FX4 : Timing data for ZPBTRF, Lower, K = 31

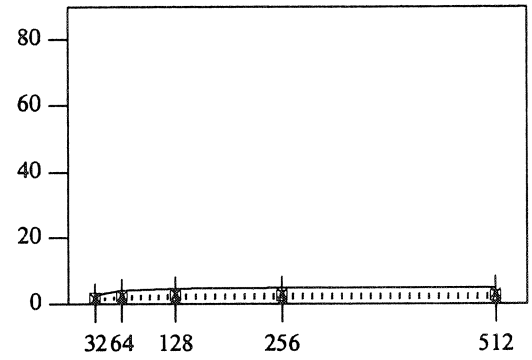

Fig.21e, 205 : Timing data for SPBTRF, Lower, K = 31

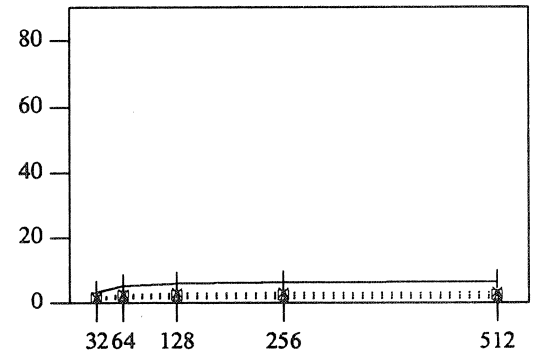

Fig.21f, 205.opt : Timing data for SPBTRF, Lower, K = 31

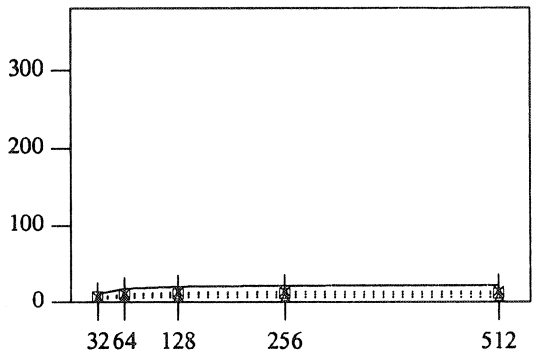

Fig.21g, NEC : Timing data for DPBTRF, Lower, K = 31

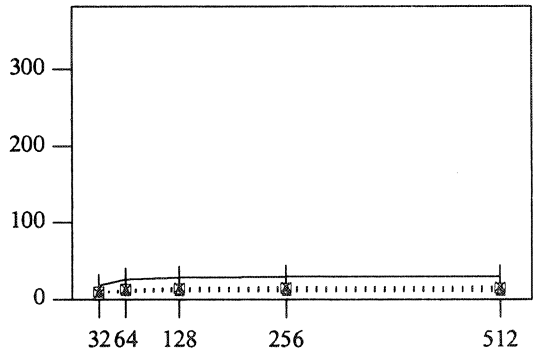

Fig.21h, NEC : Timing data for ZPBTRF, Lower, K = 31

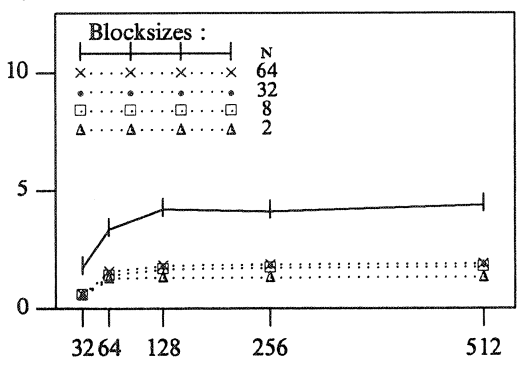

Fig.22a, FX4 : Timing data for DPBTRF, Lower, K=63

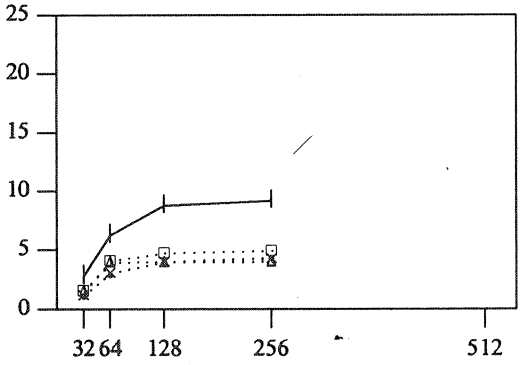

Fig.22b, 995 : Timing data for SPBTRF, Lower, K=63

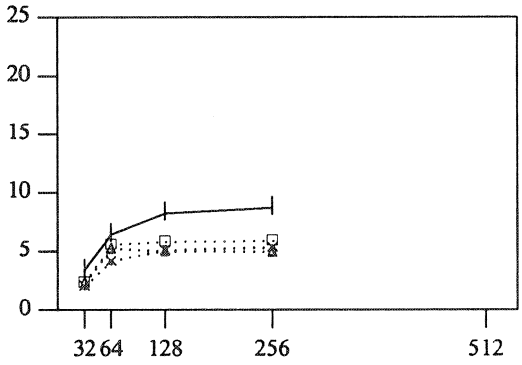

Fig.22c, IBM : Timing data for DPBTRF, Lower, K=63

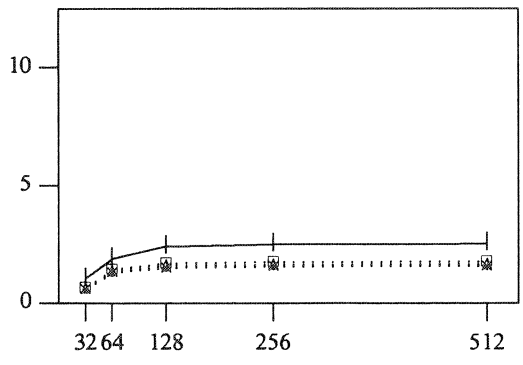

Fig.22d, FX4 : Timing data for ZPBTRF, Lower, K=63

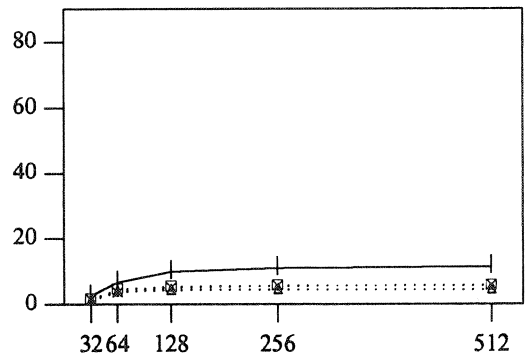

Fig.22e, 205 : Timing data for SPBTRF, Lower, K=63

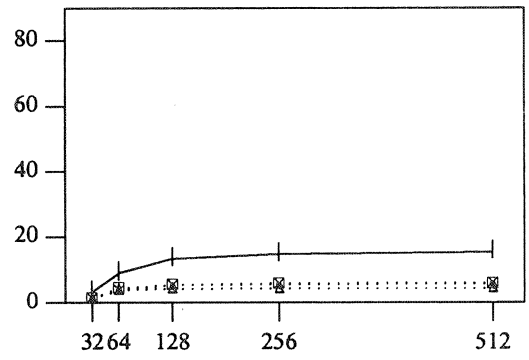

Fig.22f, 205.opt : Timing data for SPBTRF, Lower, K=63

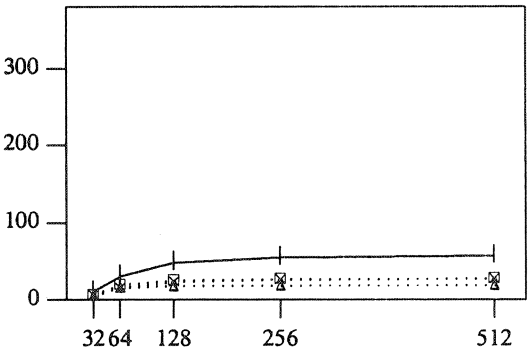

Fig.22g, NEC : Timing data for DPBTRF, Lower, K=63

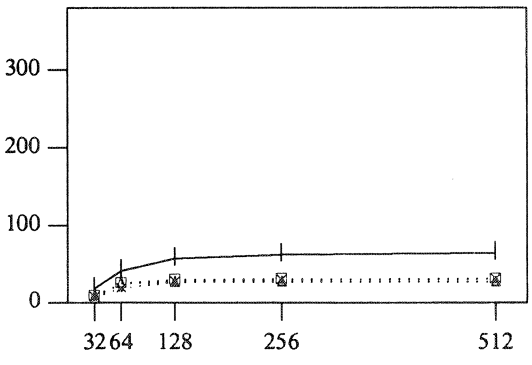

Fig.22h, NEC : Timing data for ZPBTRF, Lower, K=63

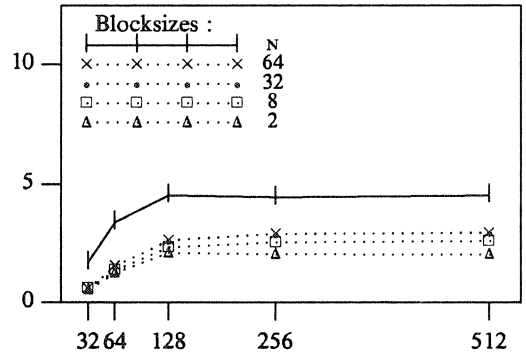

Fig.23a, FX4 : Timing data for DPBTRF, Lower, K = 127

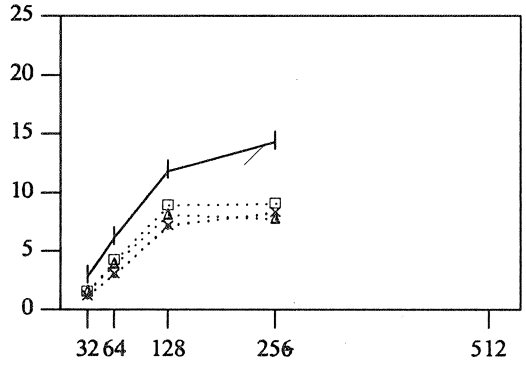

Fig.23b, 995 : Timing data for SPBTRF, Lower, K = 127

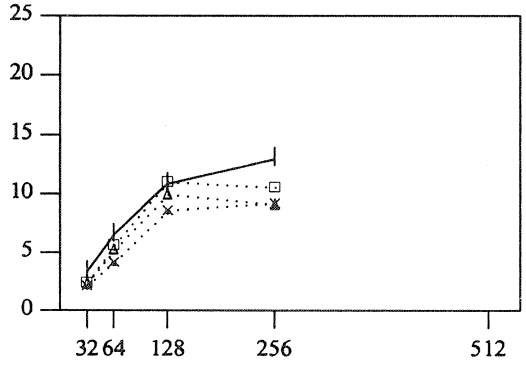

Fig.23c, IBM : Timing data for DPBTRF, Lower, K = 127

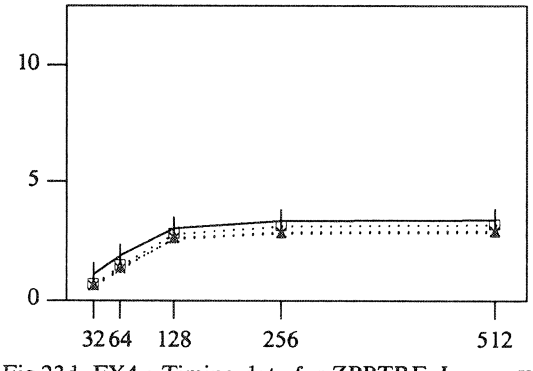

Fig.23d, FX4 : Timing data for ZPBTRF, Lower, K = 127

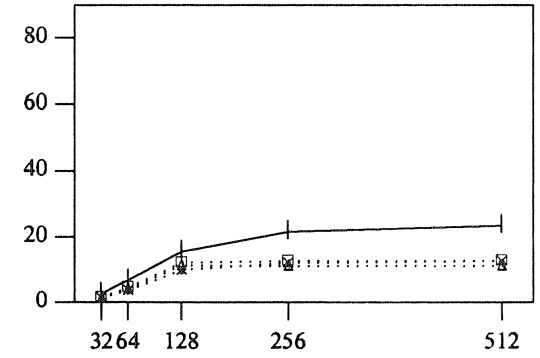

Fig.23e, 205 : Timing data for SPBTRF, Lower, K = 127

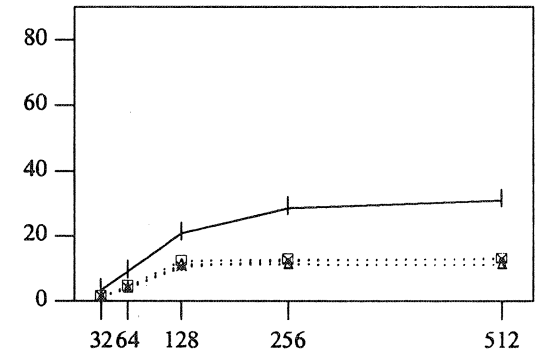

Fig.23f, 205.opt : Timing data for SPBTRF, Lower, K = 127

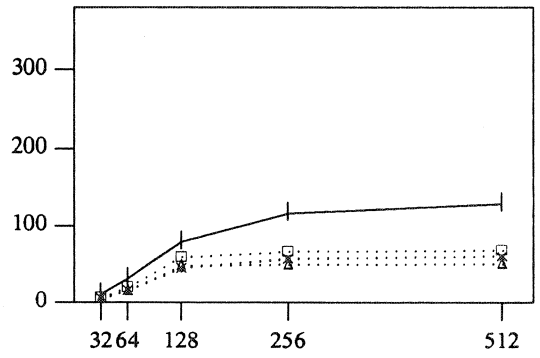

Fig.23g, NEC : Timing data for DPBTRF, Lower, K = 127

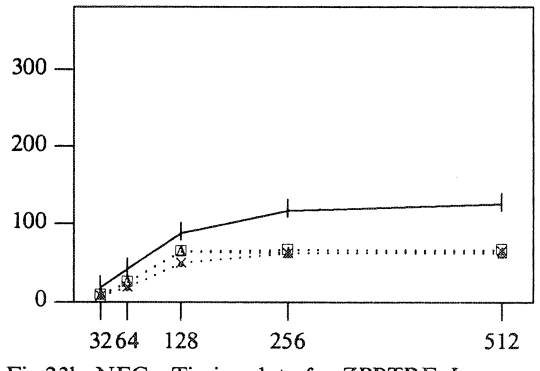

Fig.23h, NEC : Timing data for ZPBTRF, Lower, K = 127

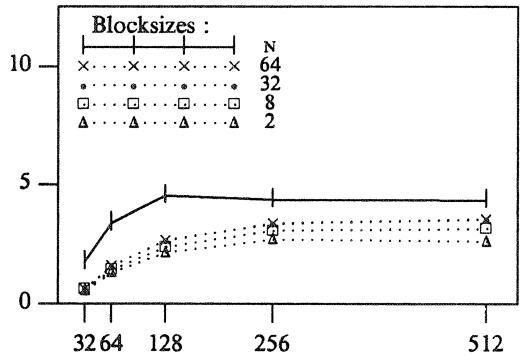

Fig.24a, FX4 : Timing data for DPBTRF, Lower, K=255

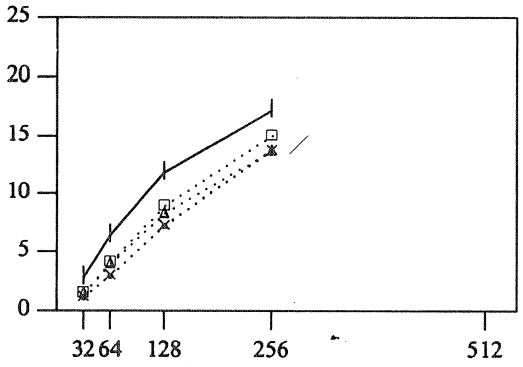

Fig.24b, 995 : Timing data for SPBTRF, Lower, K=255

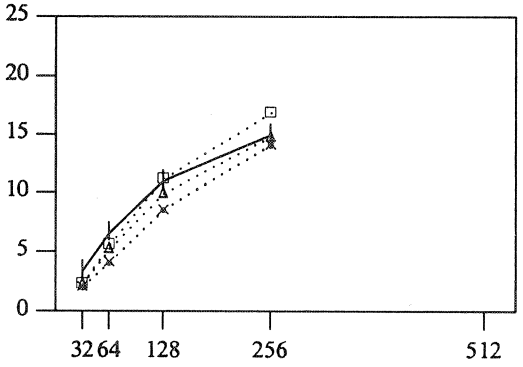

Fig.24c, IBM : Timing data for DPBTRF, Lower, K=255

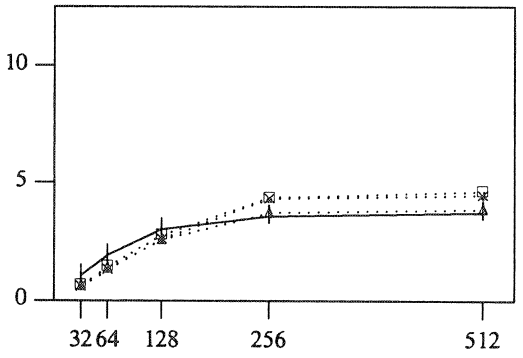

Fig.24d, FX4 : Timing data for ZPBTRF, Lower, K=255

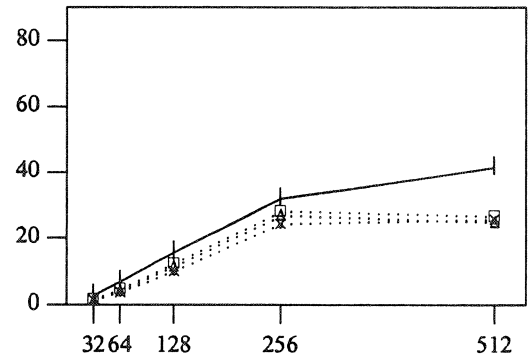

Fig.24e, 205 : Timing data for SPBTRF, Lower, K=255

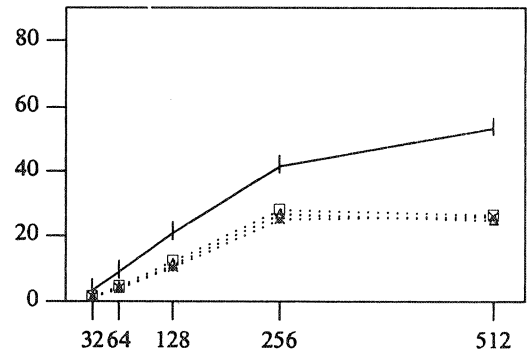

Fig.24f, 205.opt : Timing data for SPBTRF, Lower, K=255

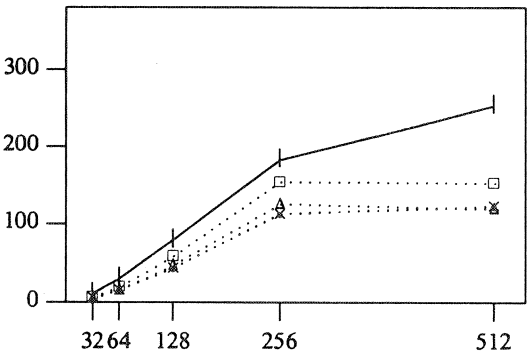

Fig.24g, NEC : Timing data for DPBTRF, Lower, K=255

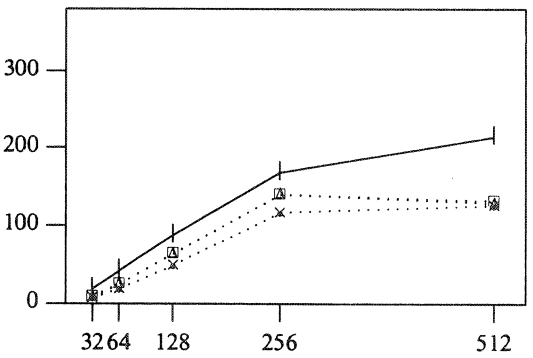

Fig.24h, NEC : Timing data for ZPBTRF, Lower, K=255

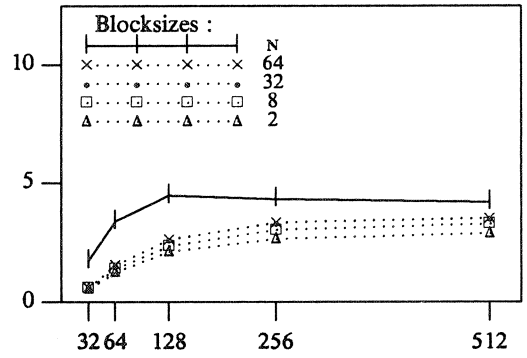

Fig.25a, FX4 : Timing data for DPBTRF, Lower, K=511

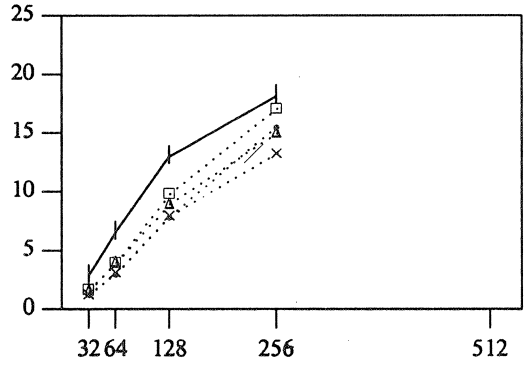

Fig.25b, 995 : Timing data for SPBTRF, Lower, K=511

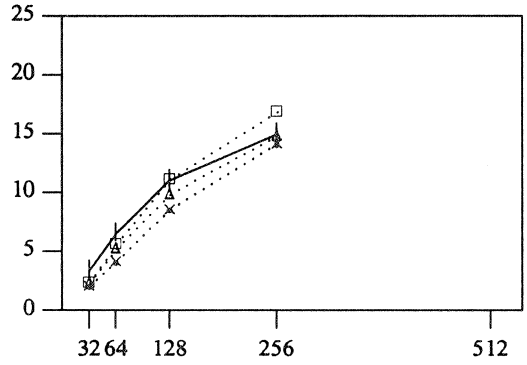

Fig.25c, IBM : Timing data for DPBTRF, Lower, K=511

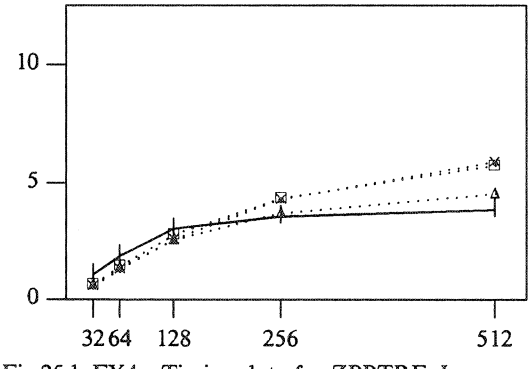

Fig.25d, FX4 : Timing data for ZPBTRF, Lower, K=511

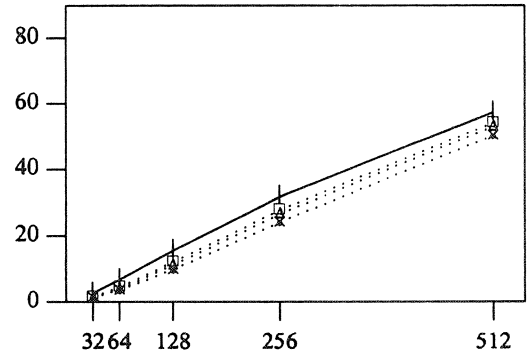

Fig.25e, 205 : Timing data for SPBTRF, Lower, K=511

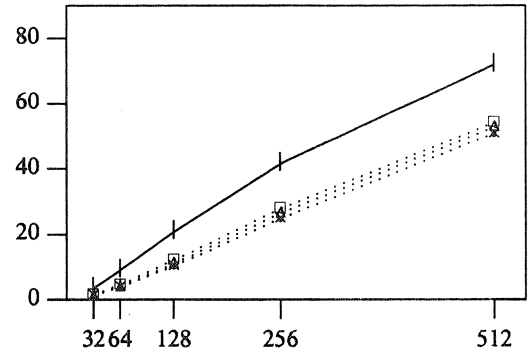

Fig.25f, 205.opt : Timing data for SPBTRF, Lower, K=511

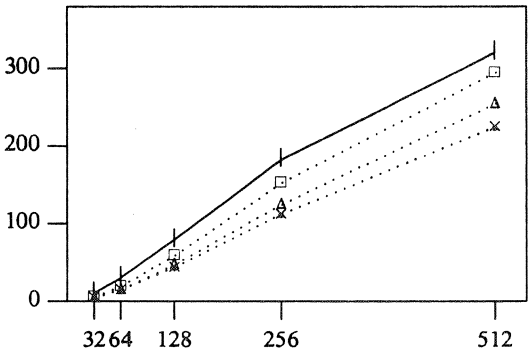

Fig.25g, NEC : Timing data for DPBTRF, Lower, K=511

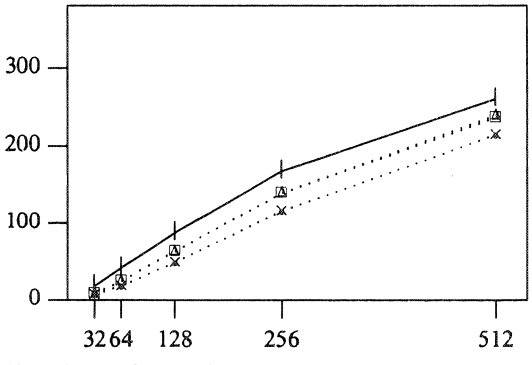

Fig.25h, NEC : Timing data for ZPBTRF, Lower, K=511

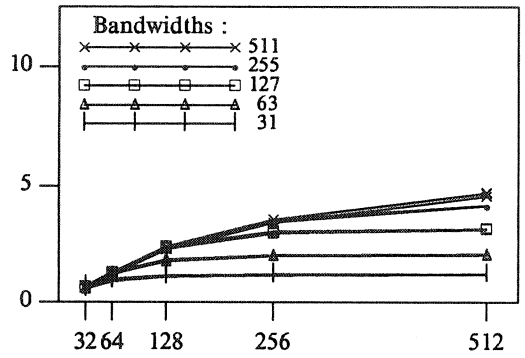

Fig.26a, FX4 : Timing data for DPBTRS, Upper

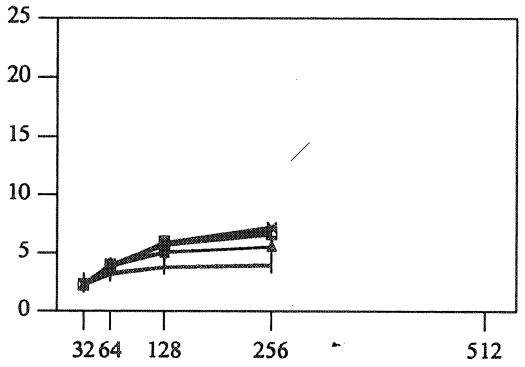

Fig.26b, 995 : Timing data for SPBTRS, Upper

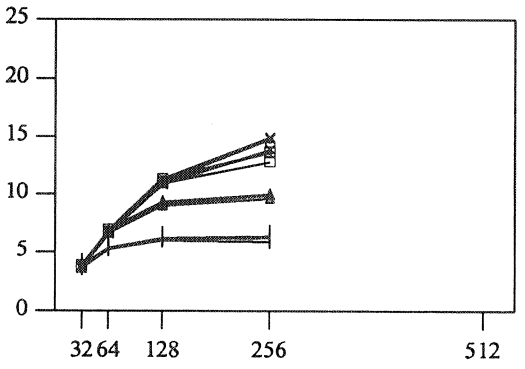

Fig.26c, IBM : Timing data for DPBTRS, Upper

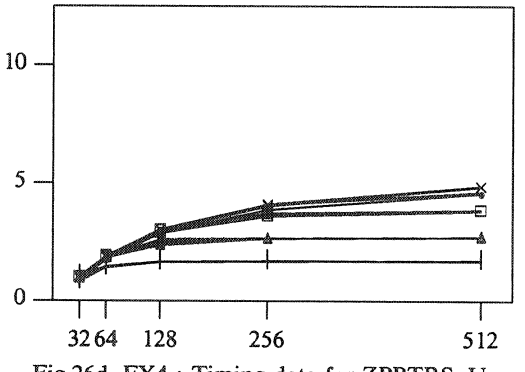

Fig.26d, FX4 : Timing data for ZPBTRS, Upper

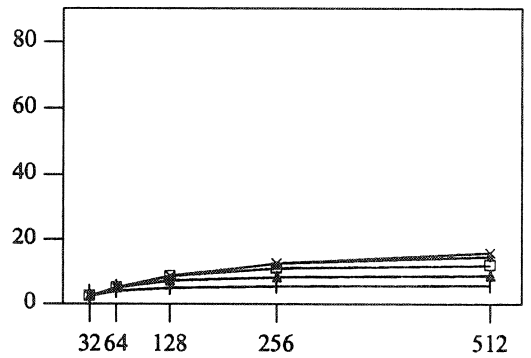

Fig.26e, 205 : Timing data for SPBTRS, Upper

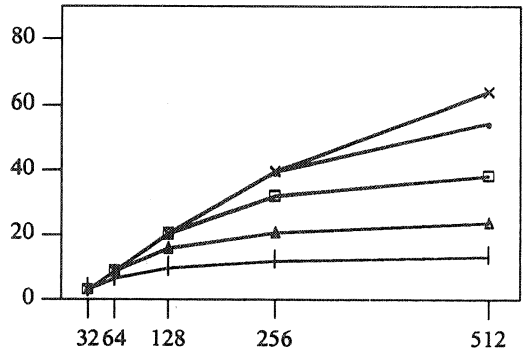

Fig.26f, 205.opt : Timing data for SPBTRS, Upper

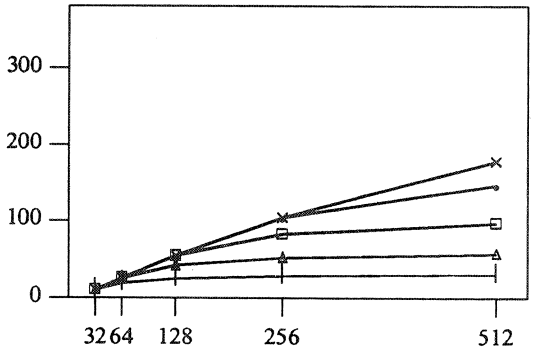

Fig.26g, NEC : Timing data for DPBTRS, Upper

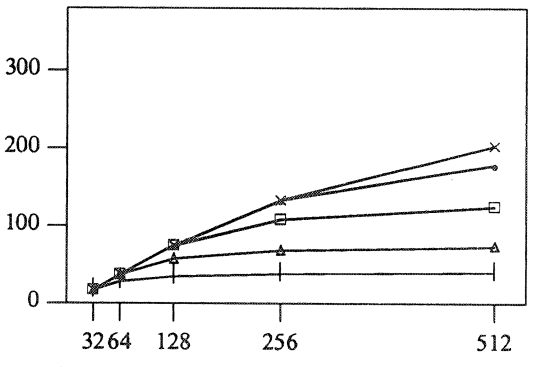

Fig.26h, NEC : Timing data for ZPBTRS, Upper

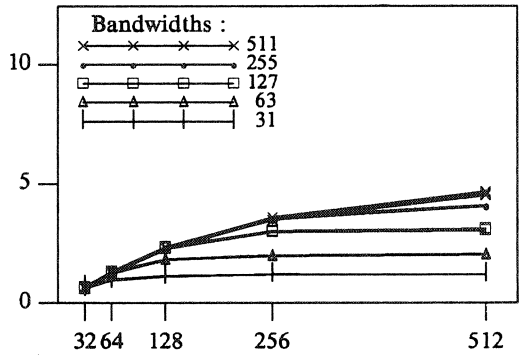

Fig.27a, FX4 : Timing data for DPBTRS, Lower

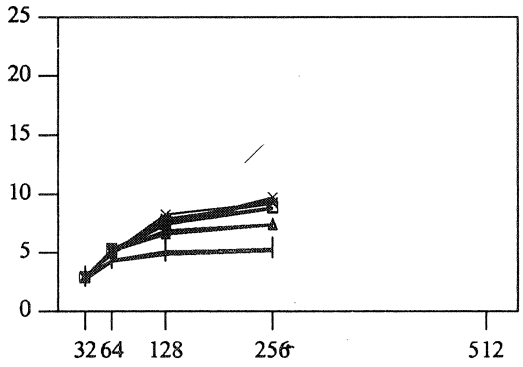

Fig.27b, 995 : Timing data for SPBTRS, Lower

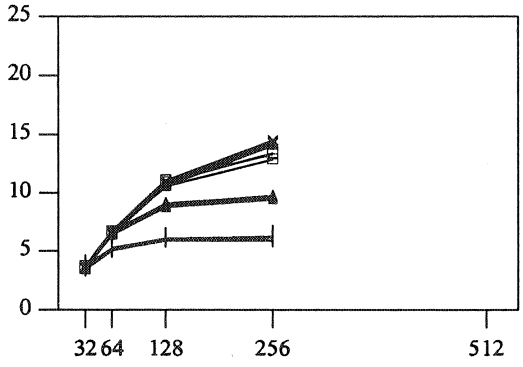

Fig.27c, IBM : Timing data for DPBTRS, Lower

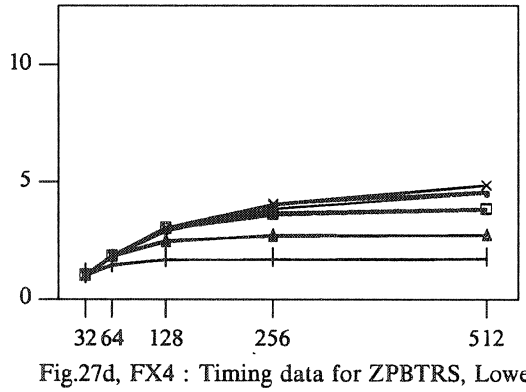

Fig.27d, FX4 : Timing data for ZPBTRS, Lower

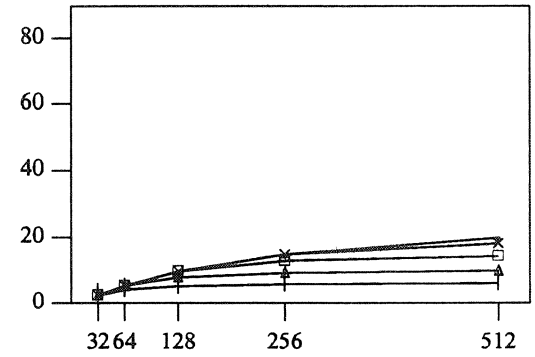

Fig.27e, 205 : Timing data for SPBTRS, Lower

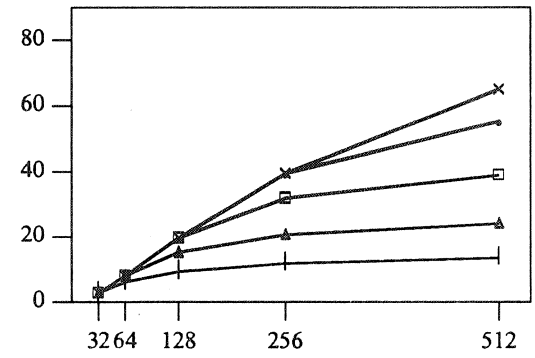

Fig.27f, 205.opt : Timing data for SPBTRS, Lower

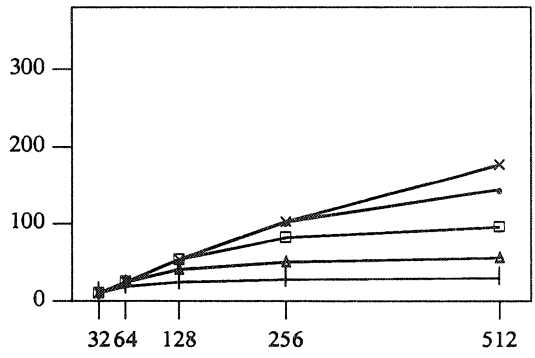

Fig.27g, NEC : Timing data for DPBTRS, Lower

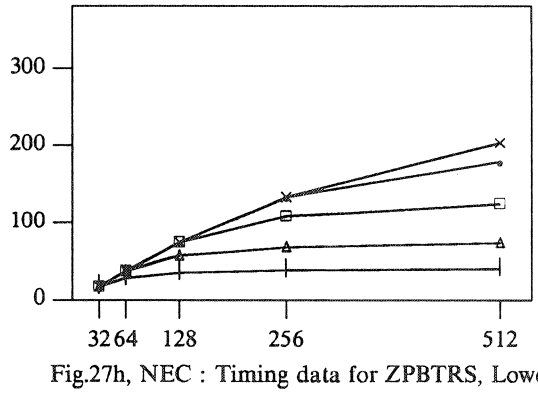

Fig.27h, NEC : Timing data for ZPBTRS, Lower

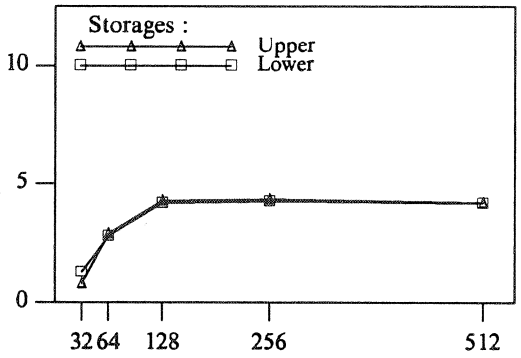

Fig.28a, FX4 : Timing data for DSYTRF

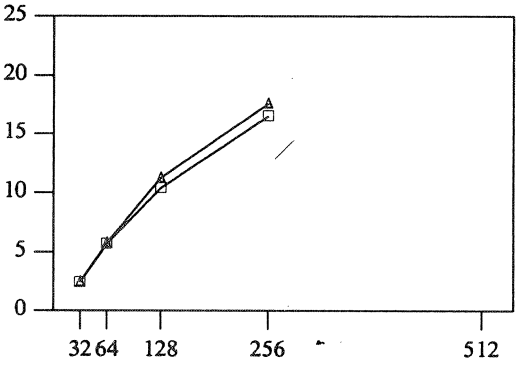

Fig.28b, 995 : Timing data for SSYTRF

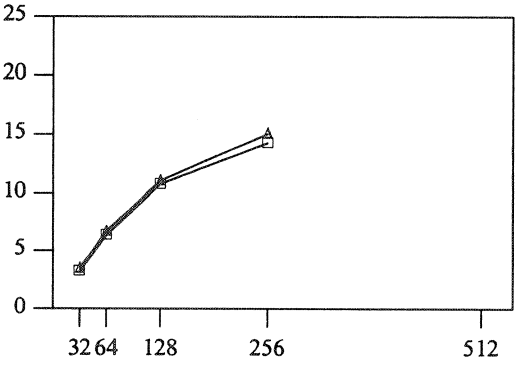

Fig.28c, IBM : Timing data for DSYTRF

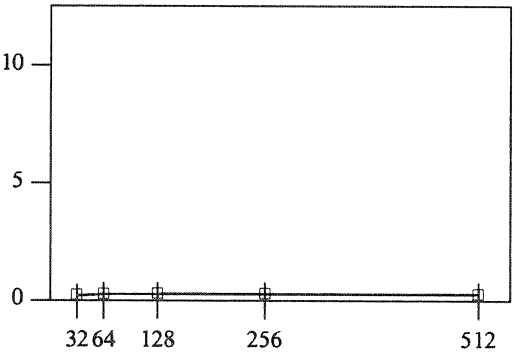

Fig.28d, FX4 : Timing data for ZSYTRF

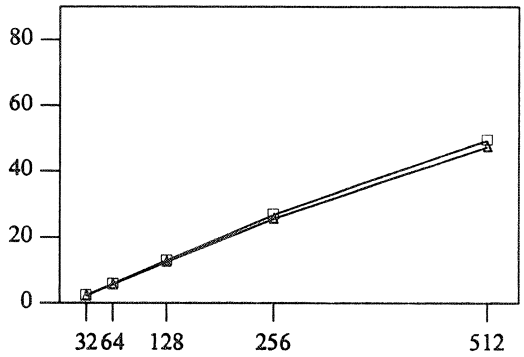

Fig.28e, 205 : Timing data for SSYTRF

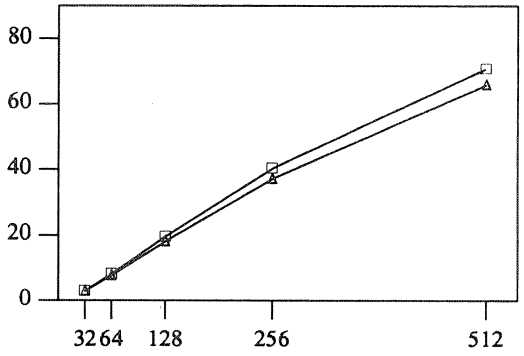

Fig.28f, 205.opt : Timing data for SSYTRF

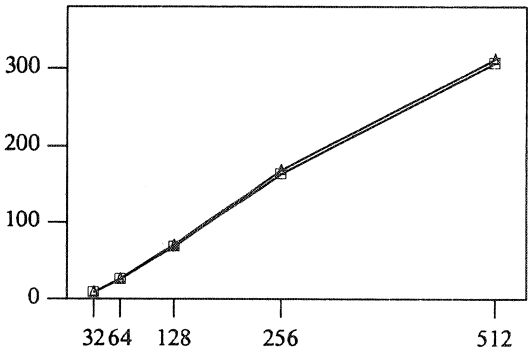

Fig.28g, NEC : Timing data for DSYTRF

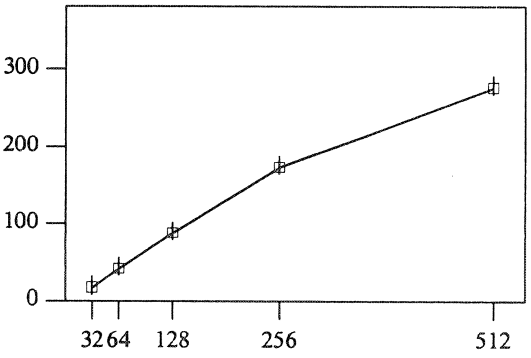

Fig.28h, NEC : Timing data for ZSYTRF

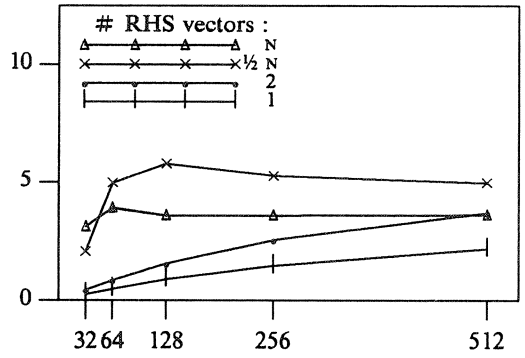

Fig.29a, FX4 : Timing data for DSYTRS, Upper

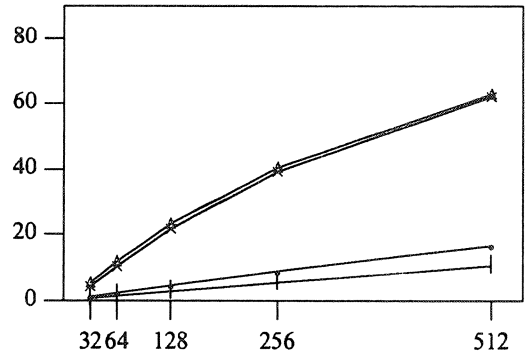

Fig.29e, 205 : Timing data for SSYTRS, Upper

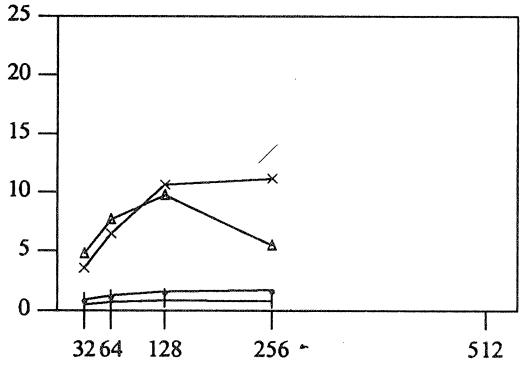

Fig.29b, 995 : Timing data for SSYTRS, Upper

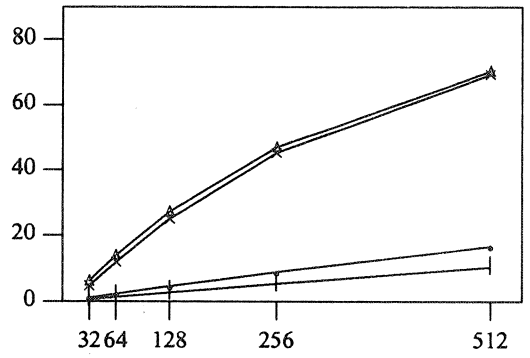

Fig.29f, 205.opt : Timing data for SSYTRS, Upper

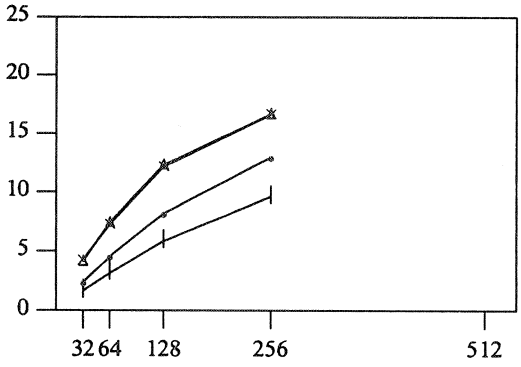

Fig.29c, IBM : Timing data for DSYTRS, Upper

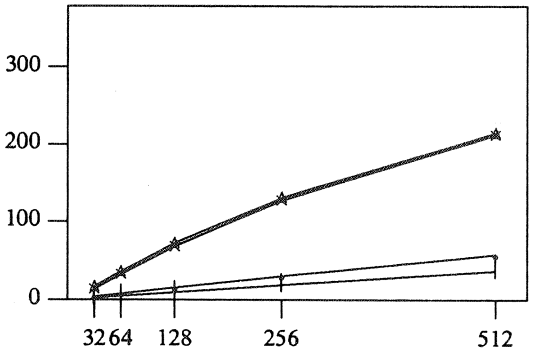

Fig.29g, NEC : Timing data for DSYTRS, Upper

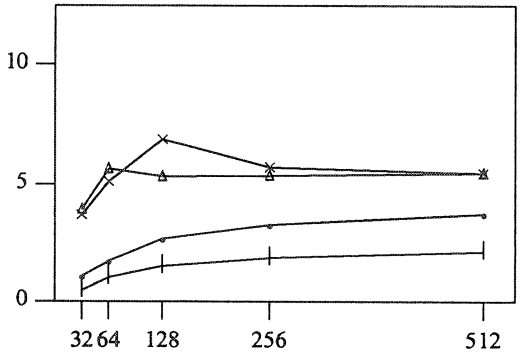

Fig.29d, FX4 : Timing data for ZSYTRS, Upper

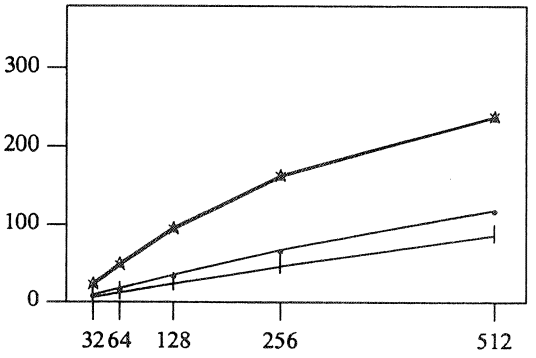

Fig.29h, NEC : Timing data for ZSYTRS, Upper

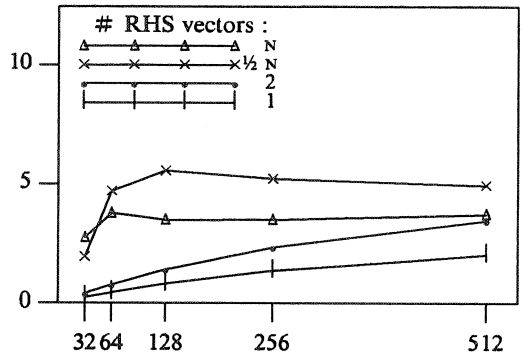

Fig.30a, FX4 : Timing data for DSYTRS, Lower

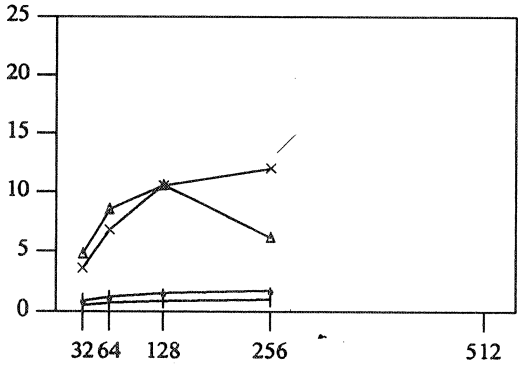

Fig.30b, 995 : Timing data for SSYTRS, Lower

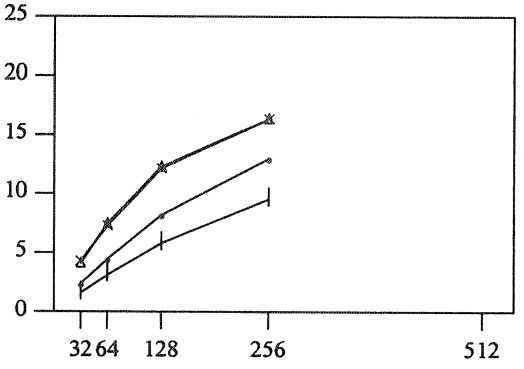

Fig.30c, IBM : Timing data for DSYTRS, Lower

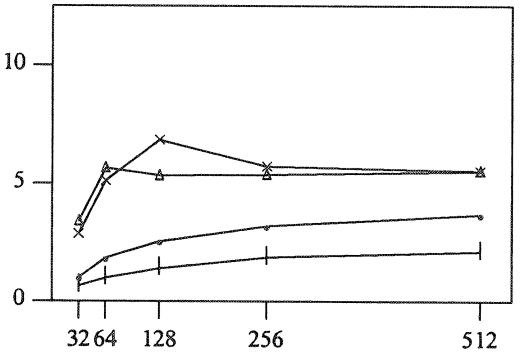

Fig.30d, FX4 : Timing data for ZSYTRS, Lower

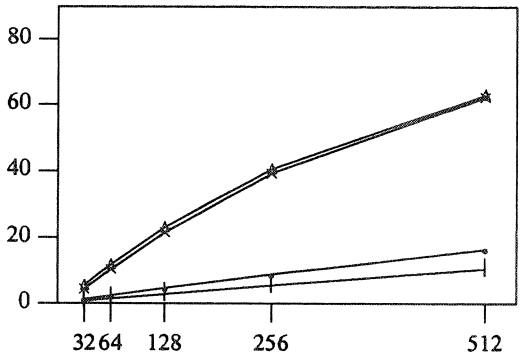

Fig.30e, 205 : Timing data for SSYTRS, Lower

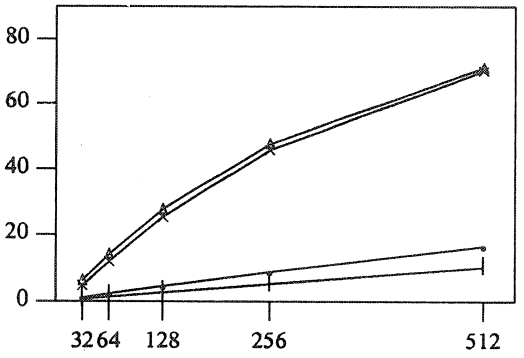

Fig.30f, 205.opt : Timing data for SSYTRS, Lower

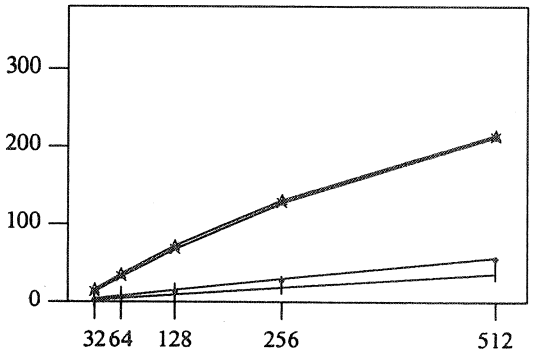

Fig.30g, NEC : Timing data for DSYTRS, Lower

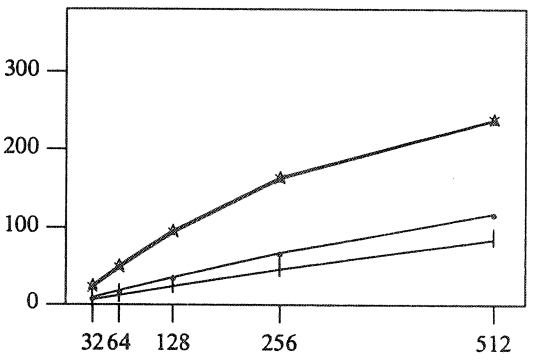

Fig.30h, NEC : Timing data for ZSYTRS, Lower

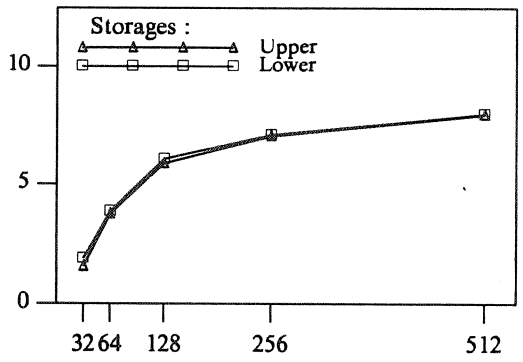

Fig.31a, FX4 : Timing data for DSYTRI

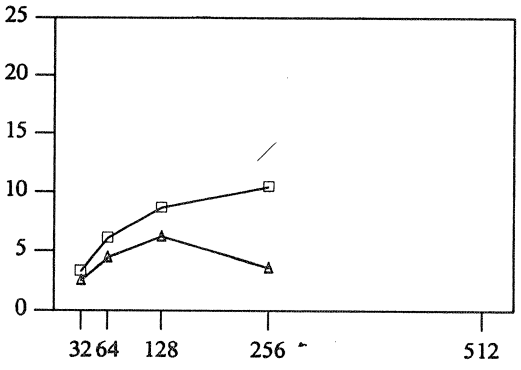

Fig.31b, 995 : Timing data for SSYTRI

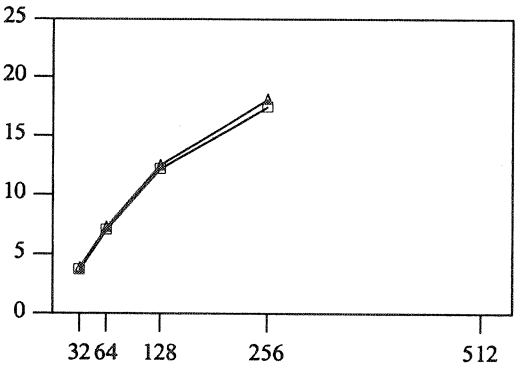

Fig.31c, IBM : Timing data for DSYTRI

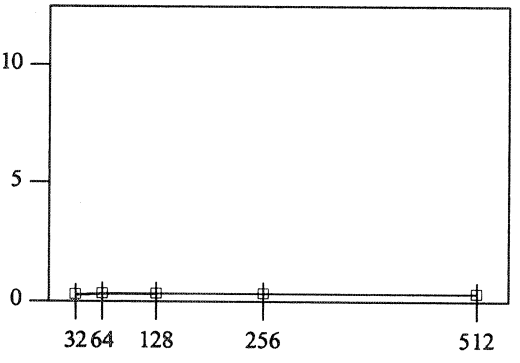

Fig.31d, FX4 : Timing data for ZSYTRI

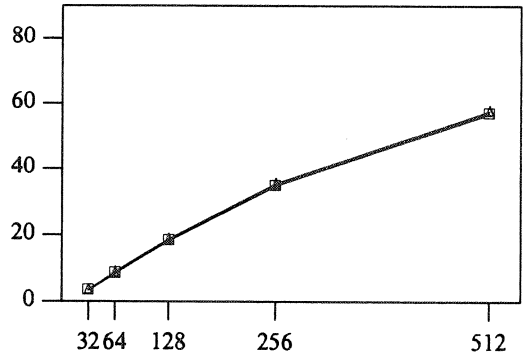

Fig.31e, 205 : Timing data for SSYTRI

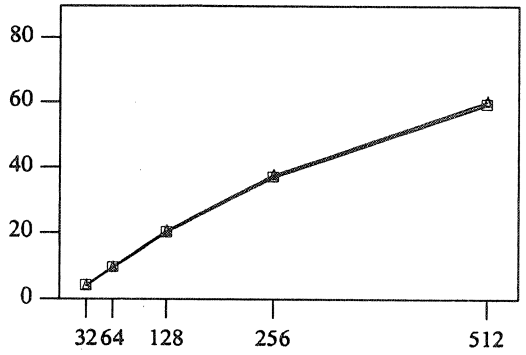

Fig.31f, 205.opt : Timing data for SSYTRI

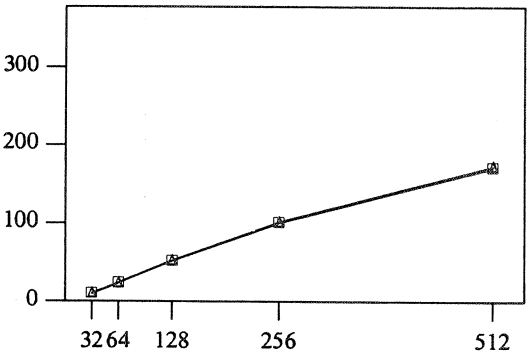

Fig.31g, NEC : Timing data for DSYTRI

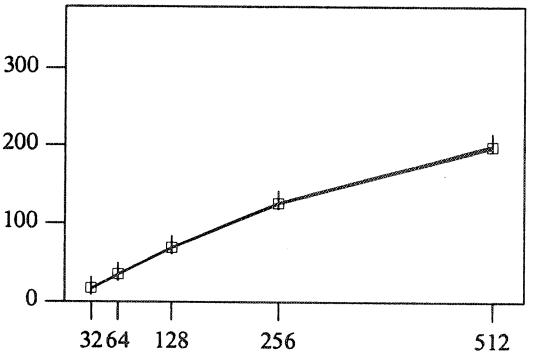

Fig.31h, NEC : Timing data for ZSYTRI

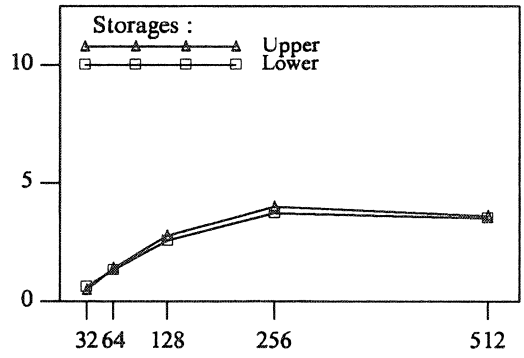

Fig.32a, FX4 : Timing data for DSPTRF

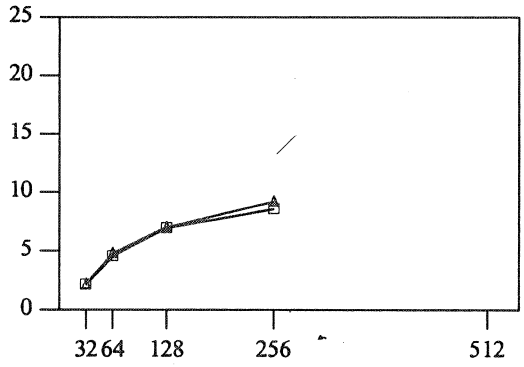

Fig.32b, 995 : Timing data for SSPTRF

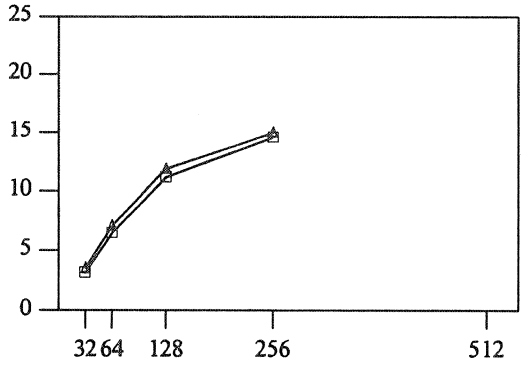

Fig.32c, IBM : Timing data for DSPTRF

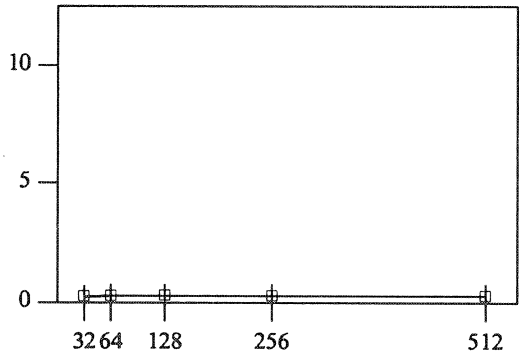

Fig.32d, FX4 : Timing data for ZSPTRF

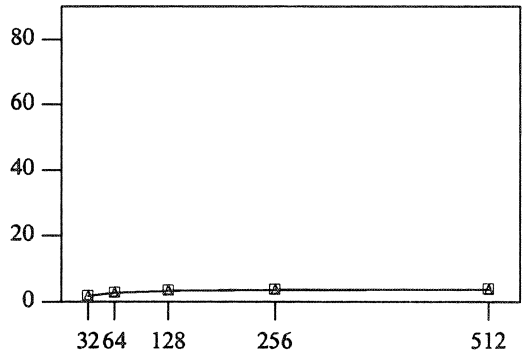

Fig.32e, 205 : Timing data for SSPTRF

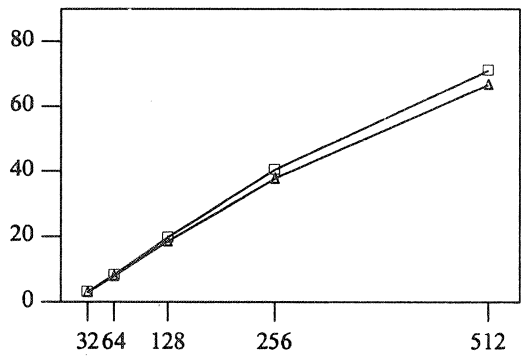

Fig.32f, 205.opt : Timing data for SSPTRF

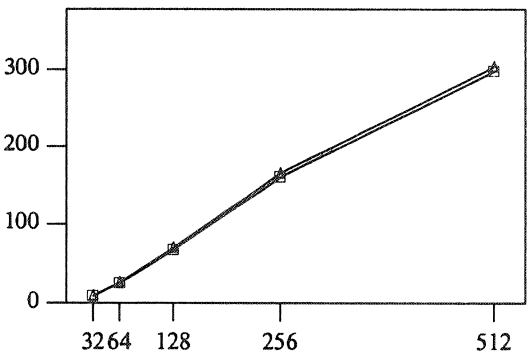

Fig.32g, NEC : Timing data for DSPTRF

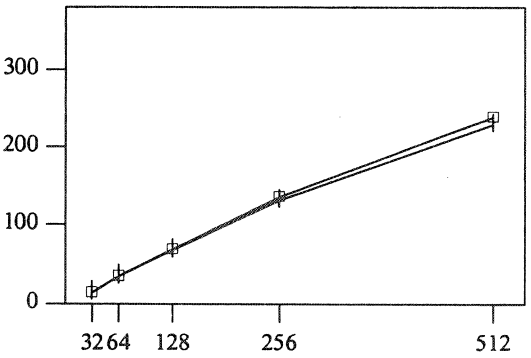

Fig.32h, NEC : Timing data for ZSPTRF

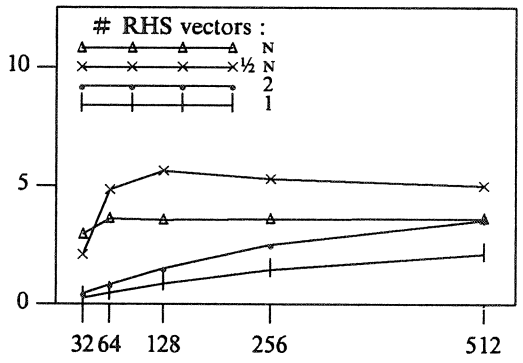

Fig.33a, FX4 : Timing data for DSPTRS, Upper

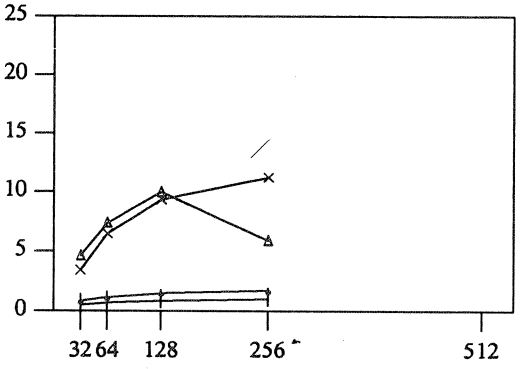

Fig.33b, 995 : Timing data for SSPTRS, Upper

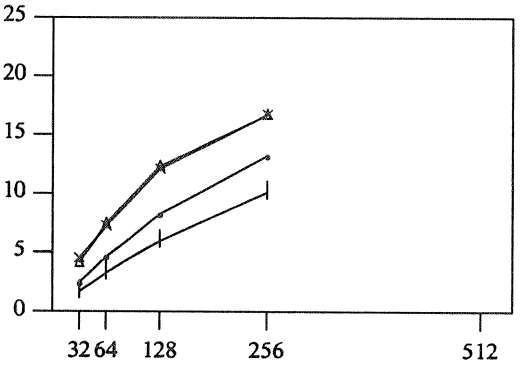

Fig.33c, IBM : Timing data for DSPTRS, Upper

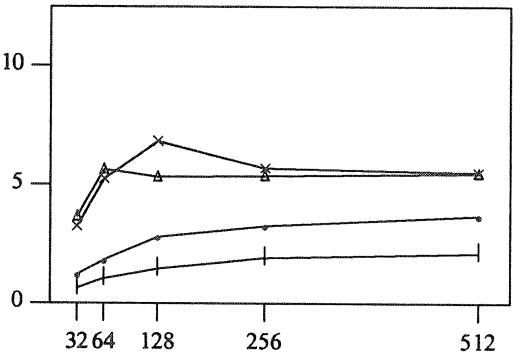

Fig.33d, FX4 : Timing data for ZSPTRS, Upper

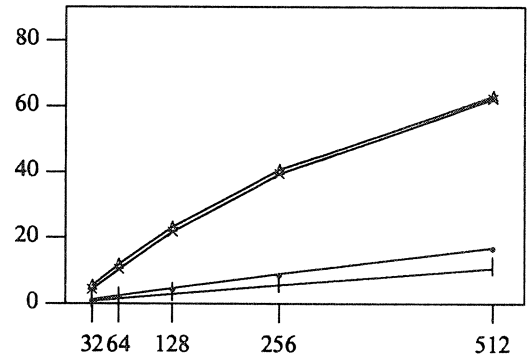

Fig.33e, 205 : Timing data for SSPTRS, Upper

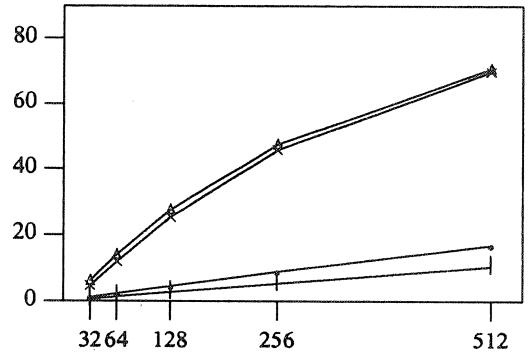

Fig.33f, 205.opt : Timing data for SSPTRS, Upper

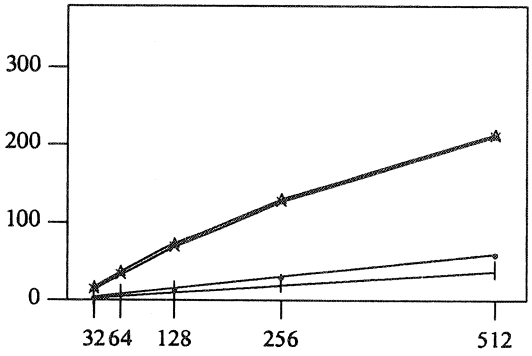

Fig.33g, NEC : Timing data for DSPTRS, Upper

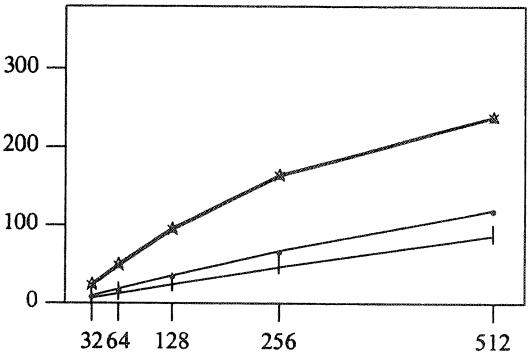

Fig.33h, NEC : Timing data for ZSPTRS, Upper

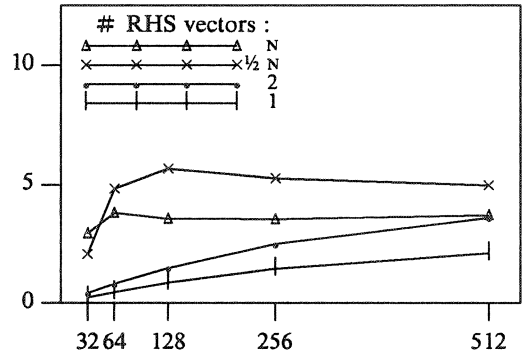

Fig.34a, FX4 : Timing data for DSPTRS, Lower

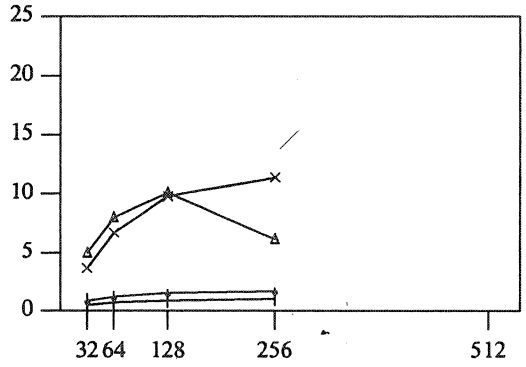

Fig.34b, 995 : Timing data for SSPTRS, Lower

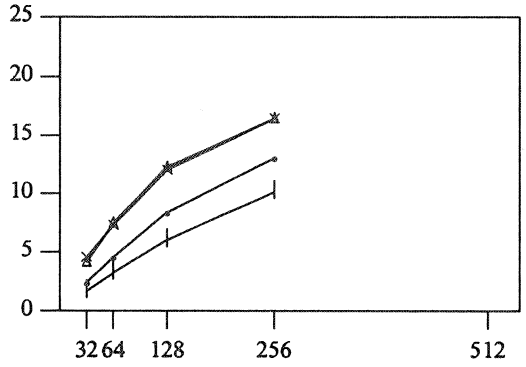

Fig.34c, IBM : Timing data for DSPTRS, Lower

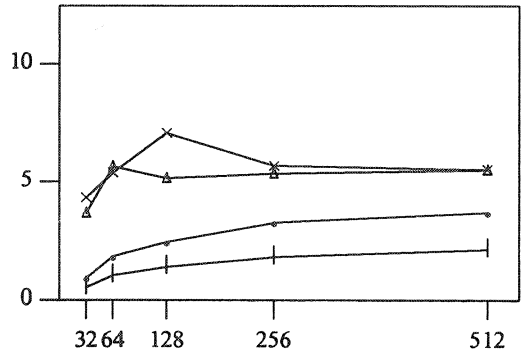

Fig.34d, FX4 : Timing data for ZSPTRS, Lower

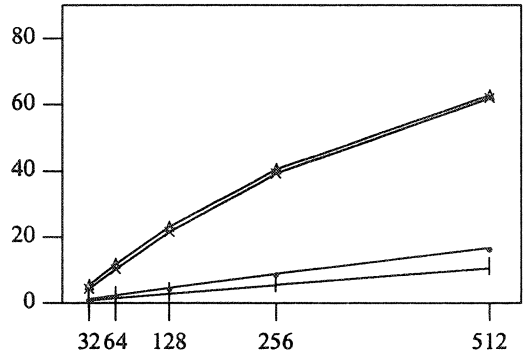

Fig.34e, 205 : Timing data for SSPTRS, Lower

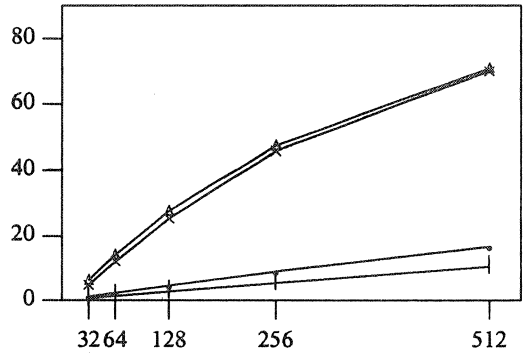

Fig.34f, 205.opt : Timing data for SSPTRS, Lower

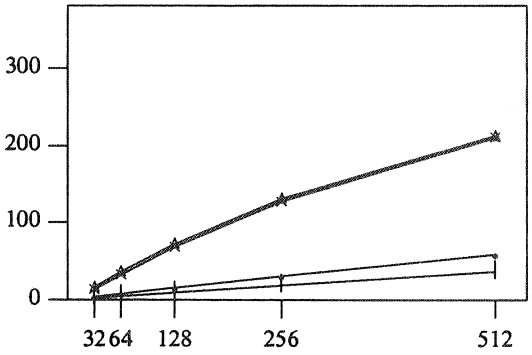

Fig.34g, NEC : Timing data for DSPTRS, Lower

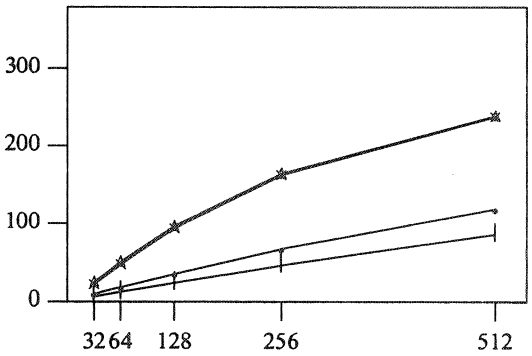

Fig.34h, NEC : Timing data for ZSPTRS, Lower

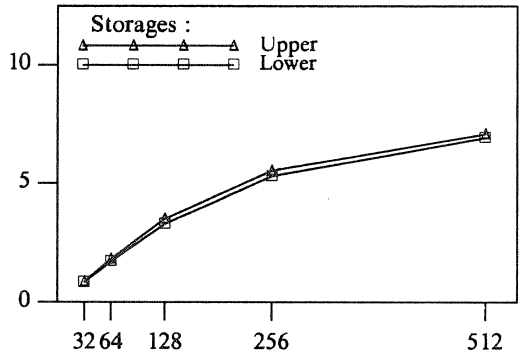

Fig.35a, FX4 : Timing data for DSPTRI

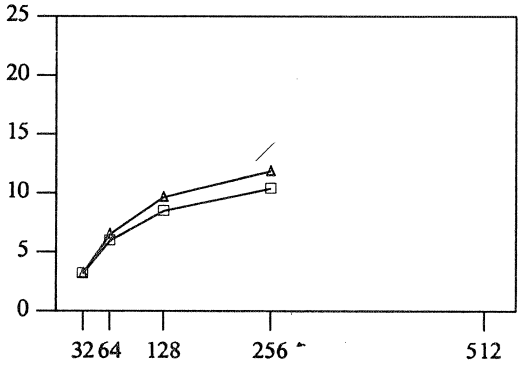

Fig.35b, 995 : Timing data for SSPTRI

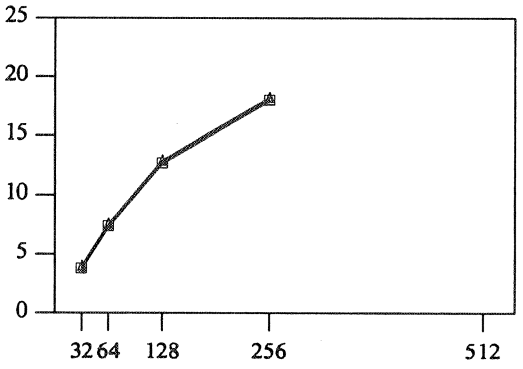

Fig.35c, IBM : Timing data for DSPTRI

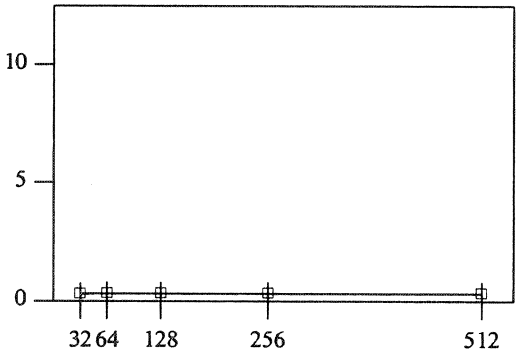

Fig.35d, FX4 : Timing data for ZSPTRI

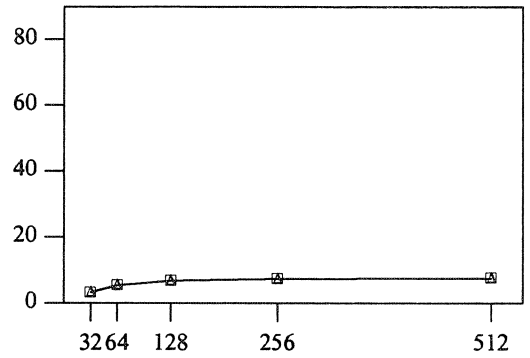

Fig.35e, 205 : Timing data for SSPTRI

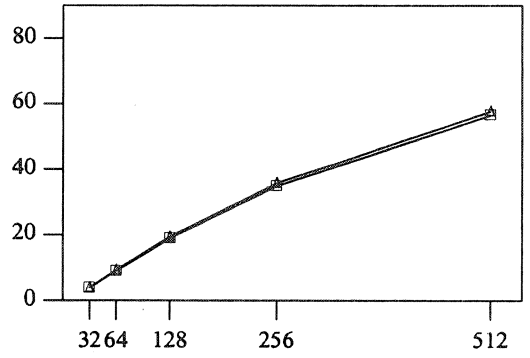

Fig.35f, 205.opt : Timing data for SSPTRI

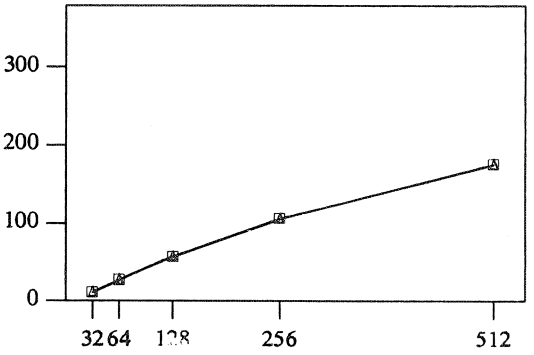

Fig.35g, NEC : Timing data for DSPTRI

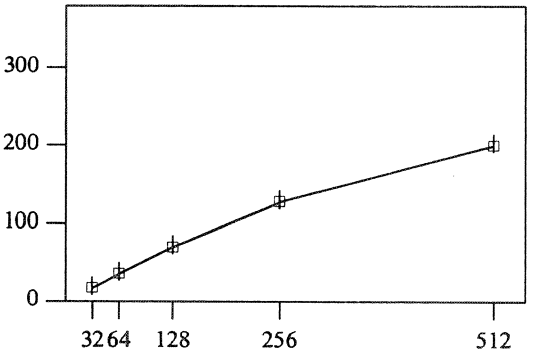

Fig.35h, NEC : Timing data for ZSPTRI

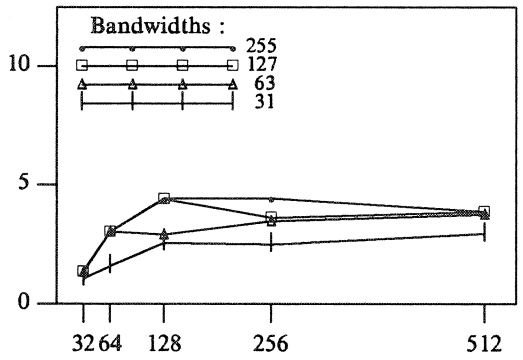

Fig.36a, FX4 : Timing data for DSBTRF, Upper

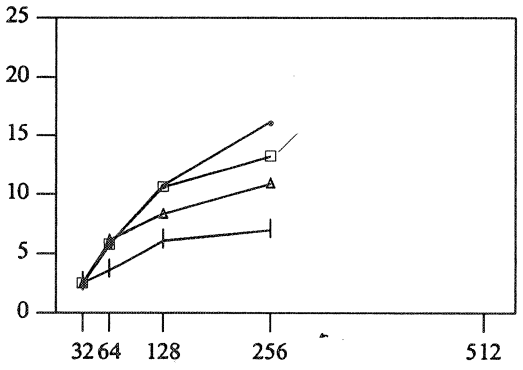

Fig.36b, 995 : Timing data for SSBTRF, Upper

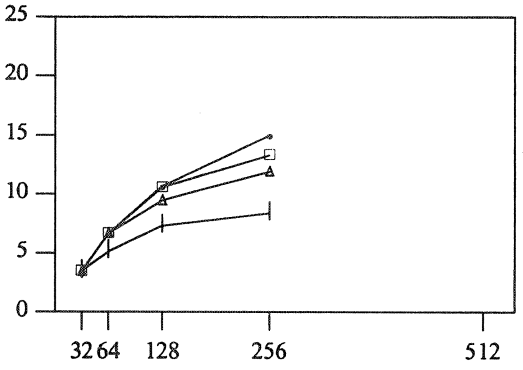

Fig.36c, IBM : Timing data for DSBTRF, Upper

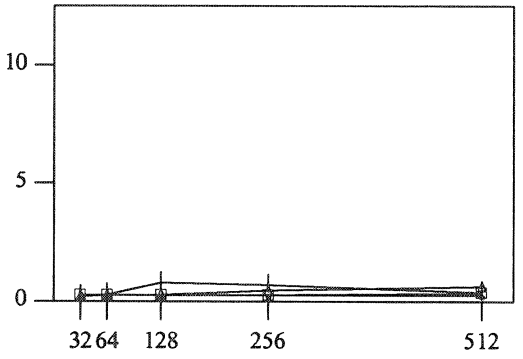

Fig.36d, FX4 : Timing data for ZSBTRF, Upper

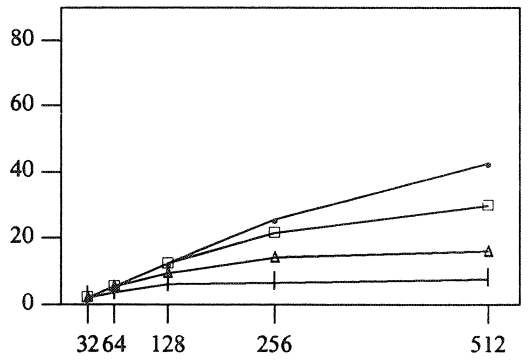

Fig.36e, 205 : Timing data for SSBTRF, Upper

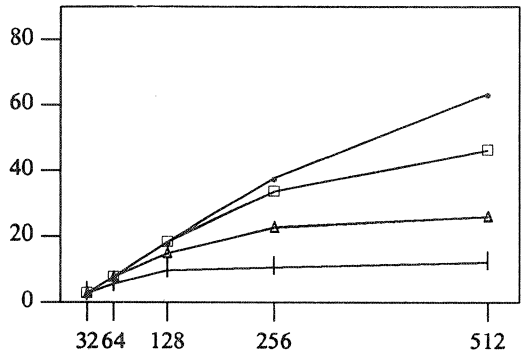

Fig.36f, 205.opt : Timing data for SSBTRF, Upper

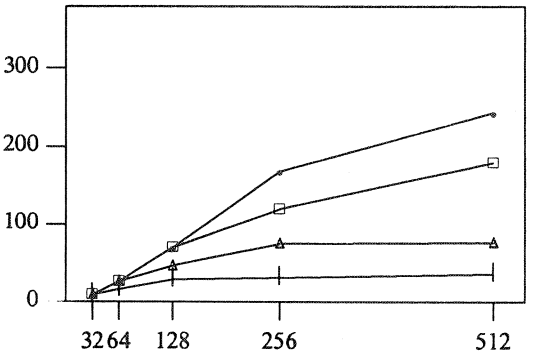

Fig.36g, NEC : Timing data for DSBTRF, Upper

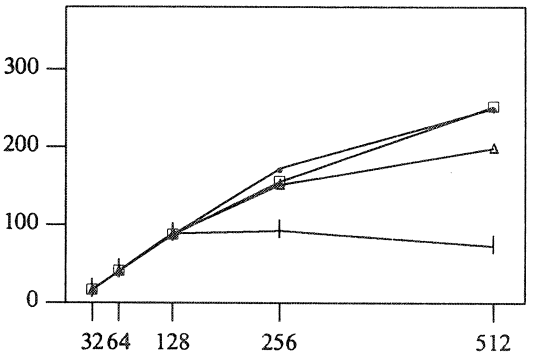

Fig.36h, NEC : Timing data for ZSBTRF, Upper

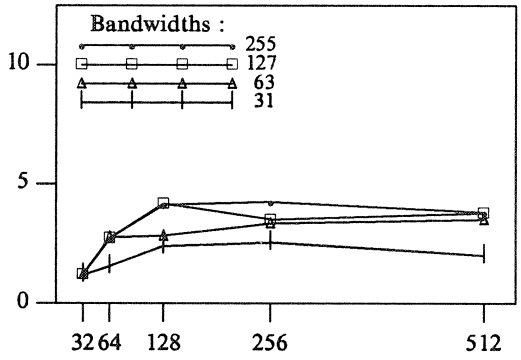

Fig.37a, FX4 : Timing data for DSBTRF, Lower

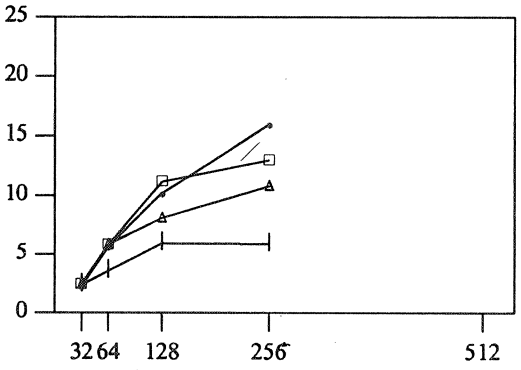

Fig.37b, 995 : Timing data for SSBTRF, Lower

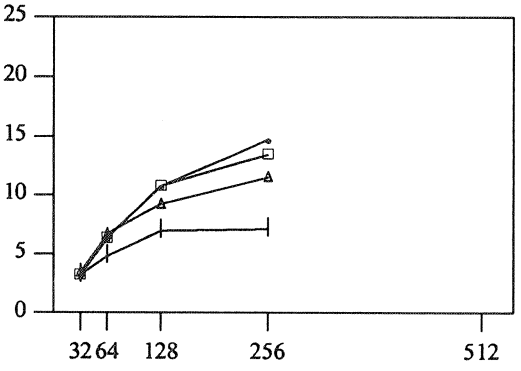

Fig.37c, IBM : Timing data for DSBTRF, Lower

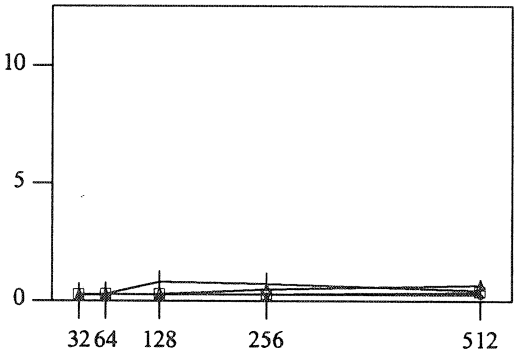

Fig.37d, FX4 : Timing data for ZSBTRF, Lower

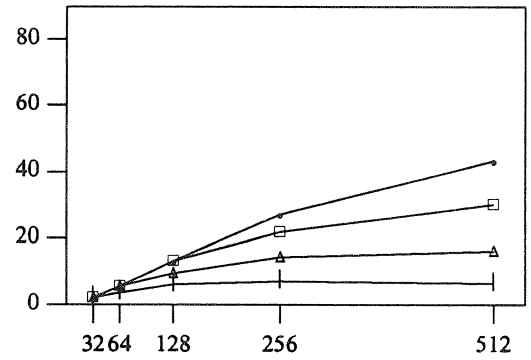

Fig.37e, 205 : Timing data for SSBTRF, Lower

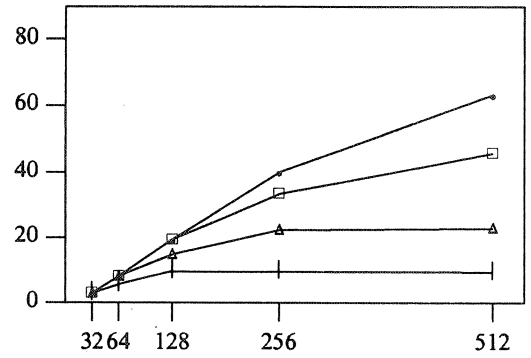

Fig.37f, 205.opt : Timing data for SSBTRF, Lower

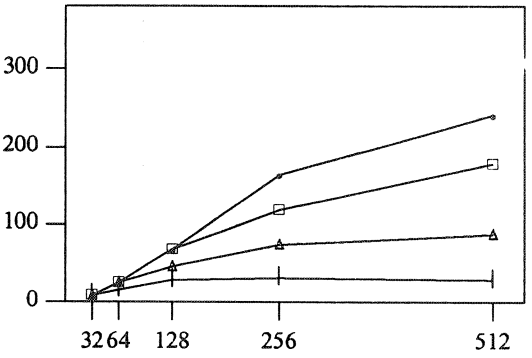

Fig.37g, NEC : Timing data for DSBTRF, Lower

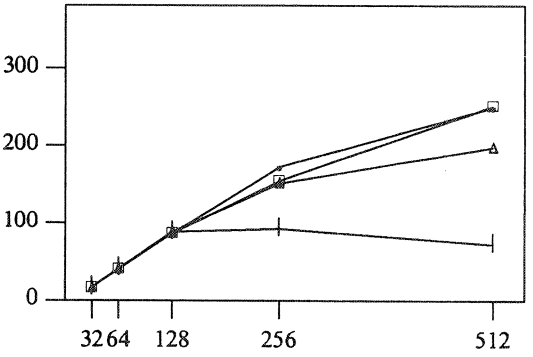

Fig.37h, NEC : Timing data for ZSBTRF, Lower

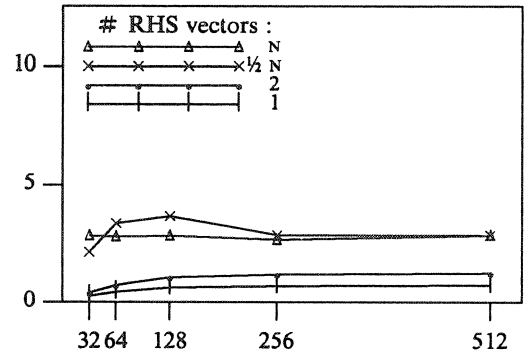

Fig.38a, FX4 : Timing data for DSBTRS, Upper, κ=31

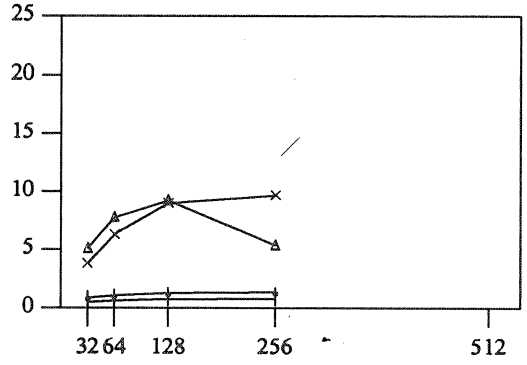

Fig.38b, 995 : Timing data for SSBTRS, Upper, κ=31

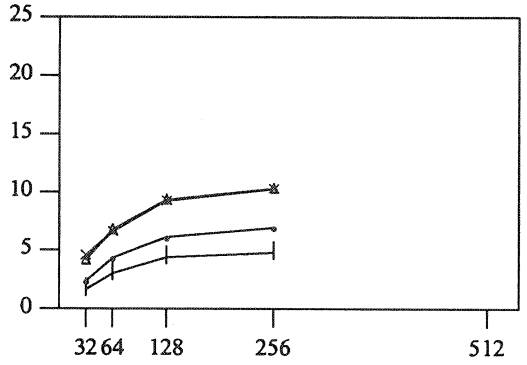

Fig.38c, IBM : Timing data for DSBTRS, Upper, κ=31

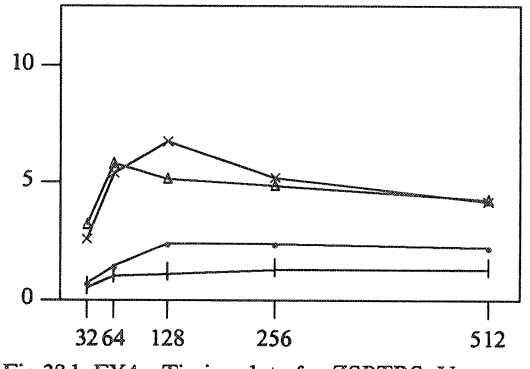

Fig.38d, FX4 : Timing data for ZSBTRS, Upper, κ=31

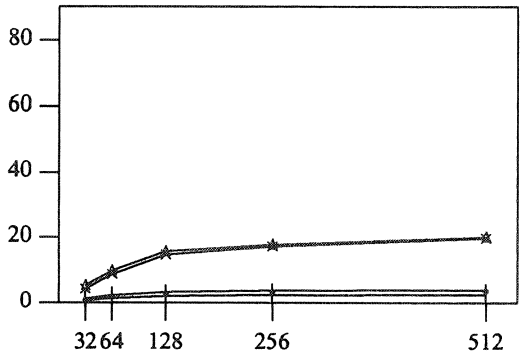

Fig.38e, 205 : Timing data for SSBTRS, Upper, κ=31

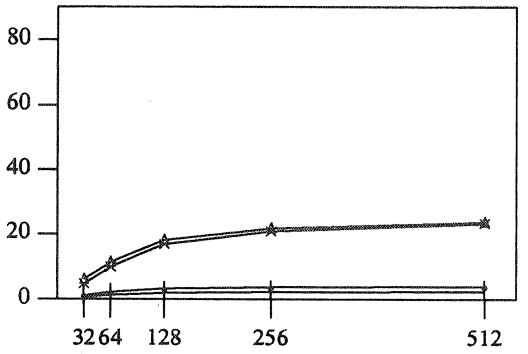

Fig.38f, 205.opt : Timing data for SSBTRS, Upper, κ=31

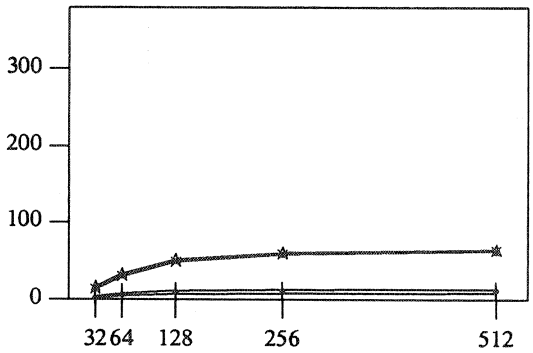

Fig.38g, NEC : Timing data for DSBTRS, Upper, κ=31

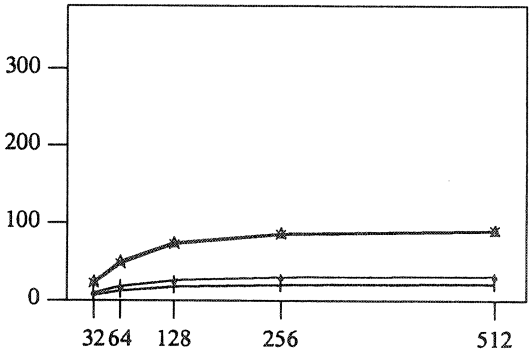

Fig.38h, NEC : Timing data for ZSBTRS, Upper, κ=31

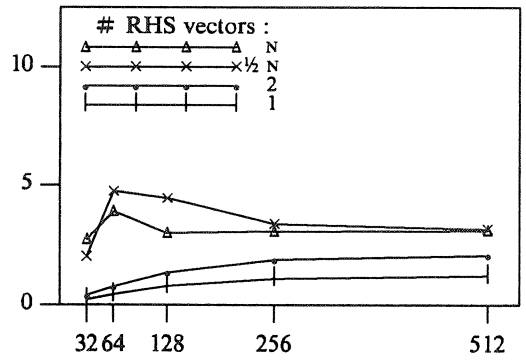

Fig.39a, FX4 : Timing data for DSBTRS, Upper, K=63

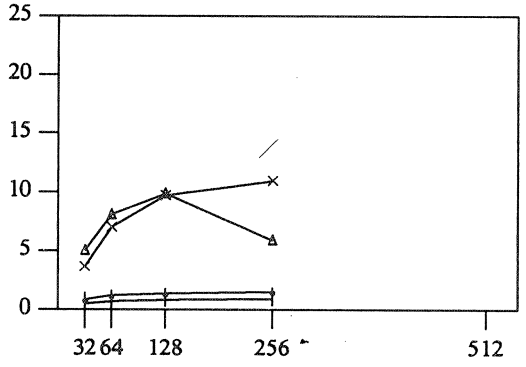

Fig.39b, 995 : Timing data for SSBTRS, Upper, K=63

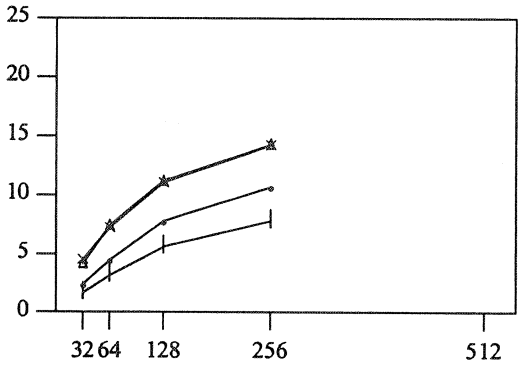

Fig.39c, IBM : Timing data for DSBTRS, Upper, K=63

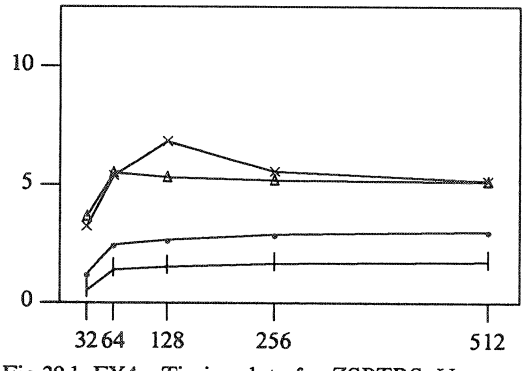

Fig.39d, FX4 : Timing data for ZSBTRS, Upper, K=63

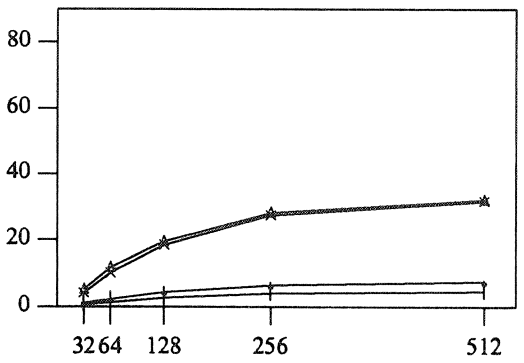

Fig.39e, 205 : Timing data for SSBTRS, Upper, K=63

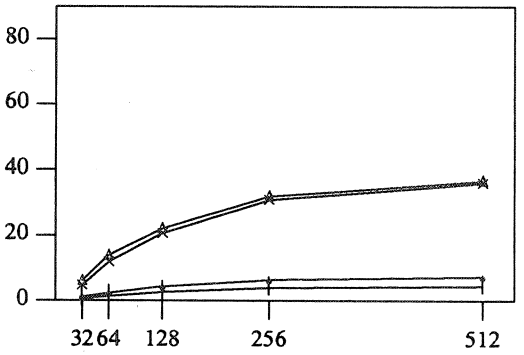

Fig.39f, 205.opt : Timing data for SSBTRS, Upper, K=63

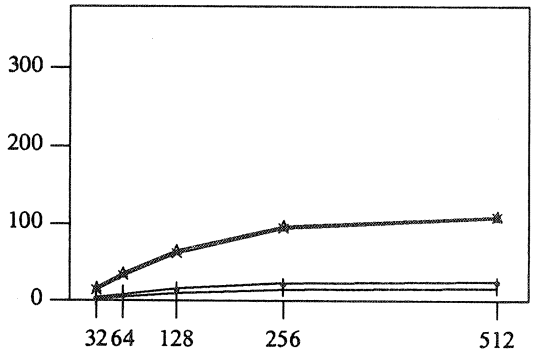

Fig.39g, NEC : Timing data for DSBTRS, Upper, K=63

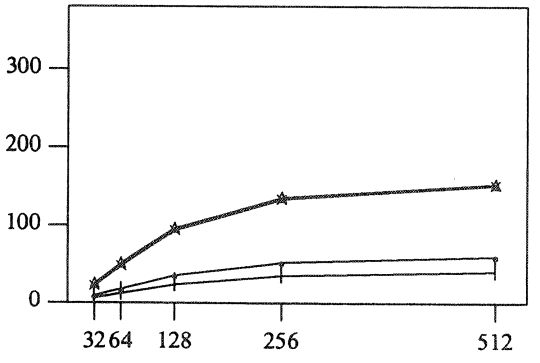

Fig.39h, NEC : Timing data for ZSBTRS, Upper, K=63

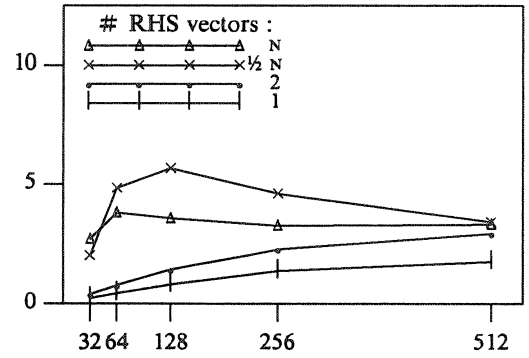

Fig.40a, FX4 : Timing data for DSBTRS, Upper, κ = 127

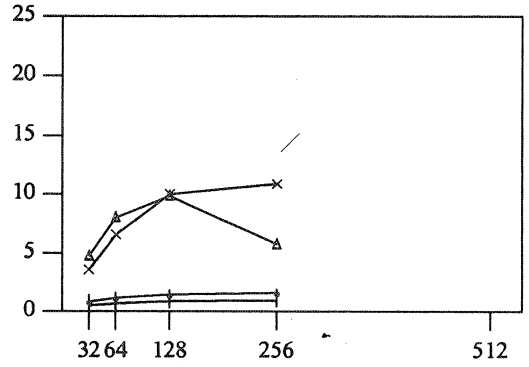

Fig.40b, 995 : Timing data for SSBTRS, Upper, κ = 127

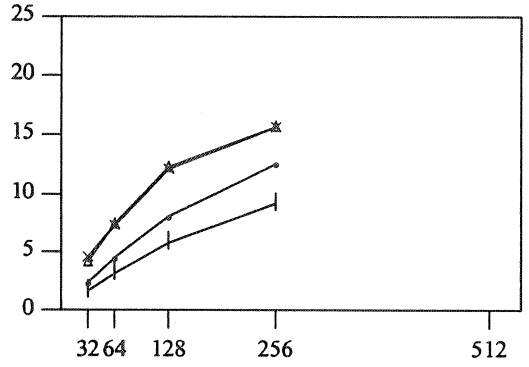

Fig.40c, IBM : Timing data for DSBTRS, Upper, κ = 127

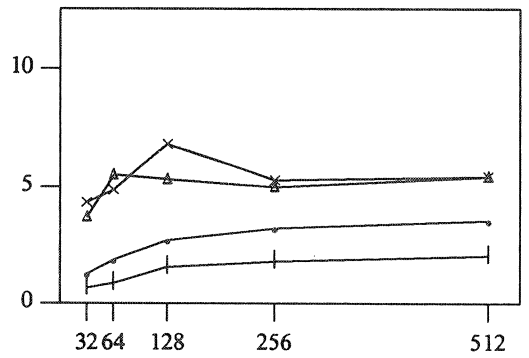

Fig.40d, FX4 : Timing data for ZSBTRS, Upper, κ = 127

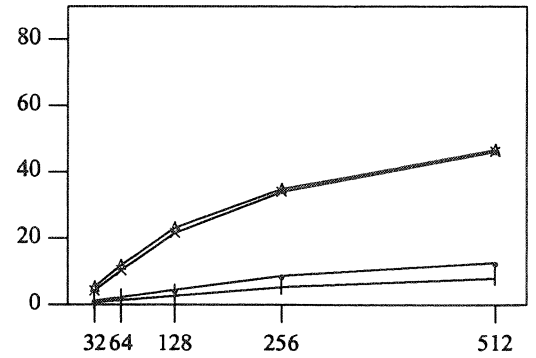

Fig.40e, 205 : Timing data for SSBTRS, Upper, κ = 127

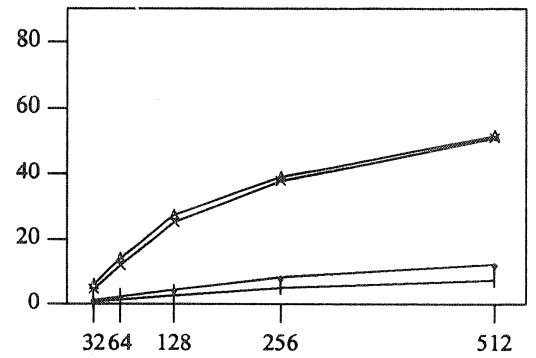

Fig.40f, 205.opt : Timing data for SSBTRS, Upper, κ = 127

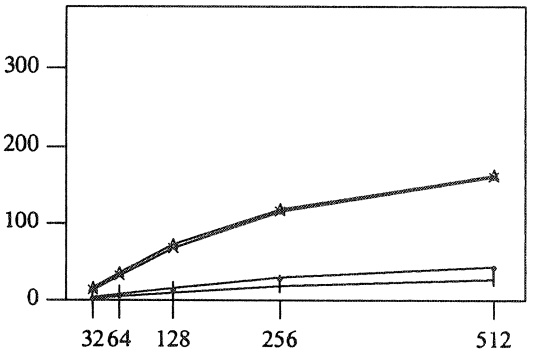

Fig.40g, NEC : Timing data for DSBTRS, Upper, κ = 127

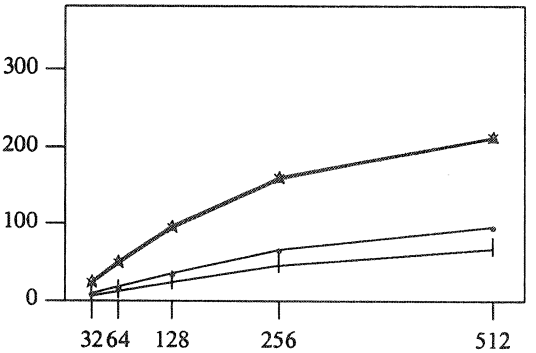

Fig.40h, NEC : Timing data for ZSBTRS, Upper, κ = 127

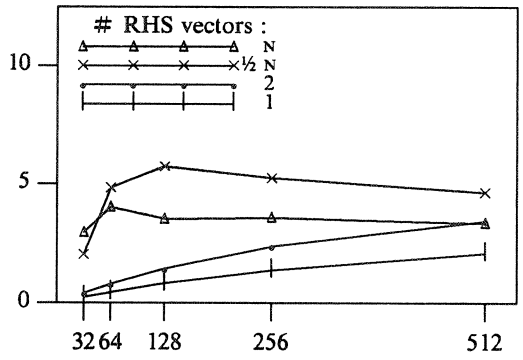

Fig.41a, FX4 : Timing data for DSBTRS, Upper, κ=255

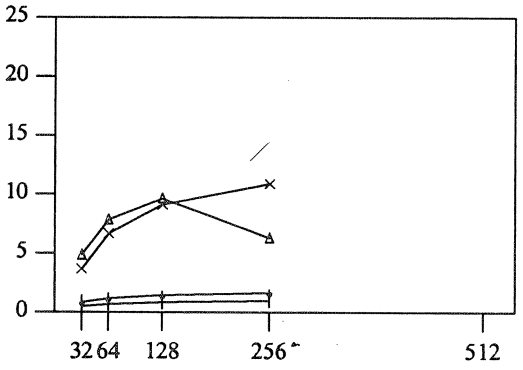

Fig.41b, 995 : Timing data for SSBTRS, Upper, κ=255

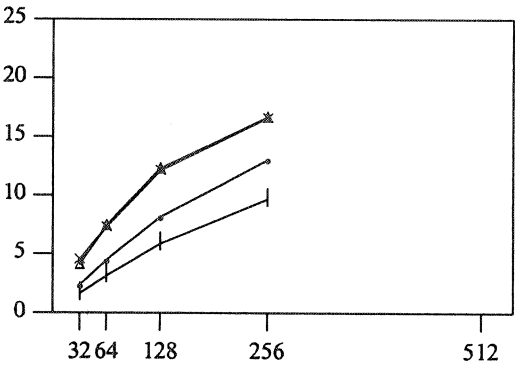

Fig.41c, IBM : Timing data for DSBTRS, Upper, κ=255

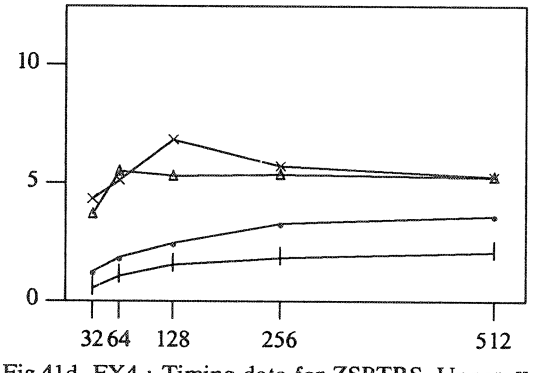

Fig.41d, FX4 : Timing data for ZSBTRS, Upper, κ=255

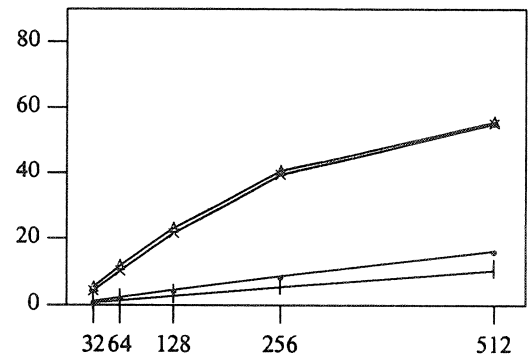

Fig.41e, 205 : Timing data for SSBTRS, Upper, κ=255

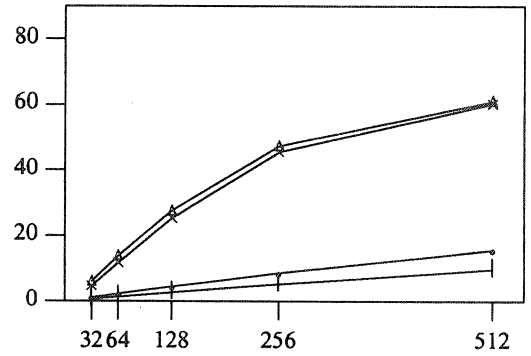

Fig.41f, 205.opt : Timing data for SSBTRS, Upper, κ=255

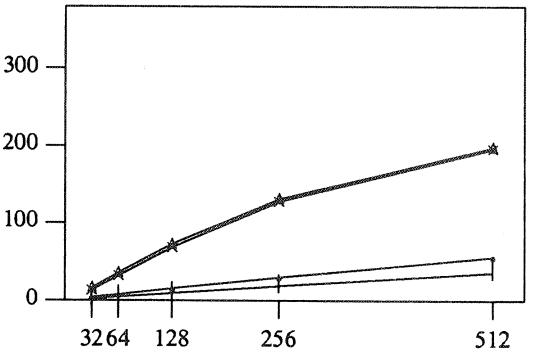

Fig.41g, NEC : Timing data for DSBTRS, Upper, κ=255

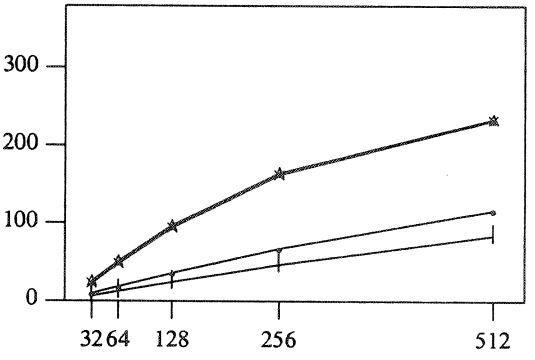

Fig.41h, NEC : Timing data for ZSBTRS, Upper, κ=255

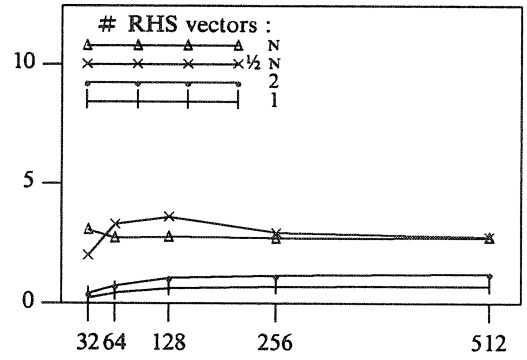

Fig.42a, FX4 : Timing data for DSBTRS, Lower, κ=31

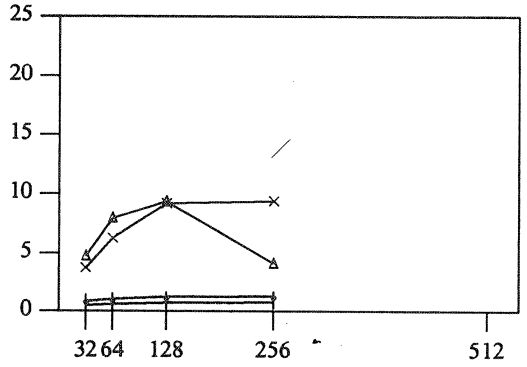

Fig.42b, 995 : Timing data for SSBTRS, Lower, κ=31

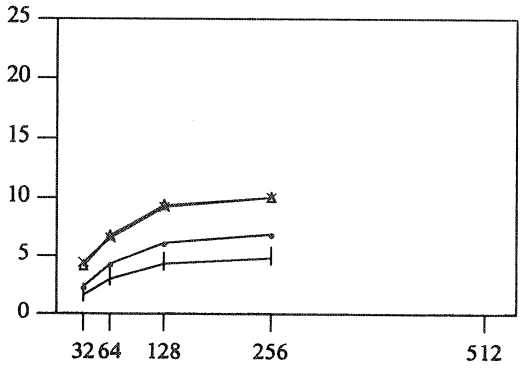

Fig.42c, IBM : Timing data for DSBTRS, Lower, κ=31

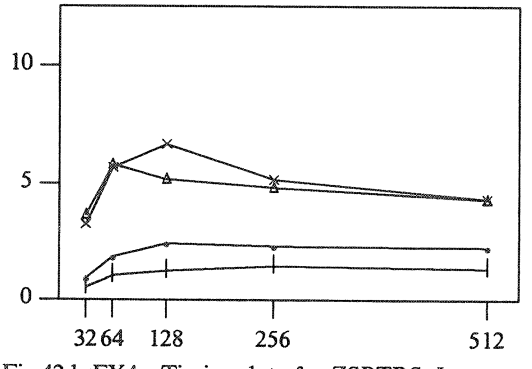

Fig.42d, FX4 : Timing data for ZSBTRS, Lower, κ=31

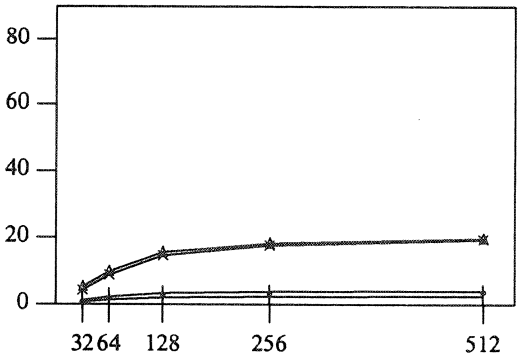

Fig.42e, 205 : Timing data for SSBTRS, Lower, κ=31

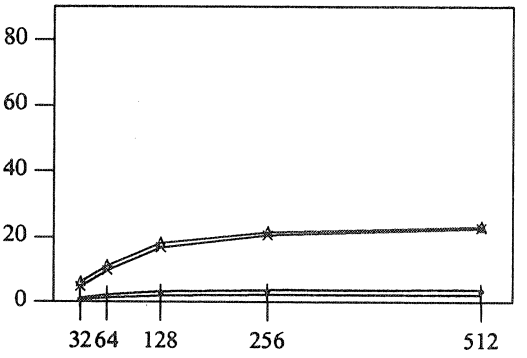

Fig.42f, 205.opt : Timing data for SSBTRS, Lower, κ=31

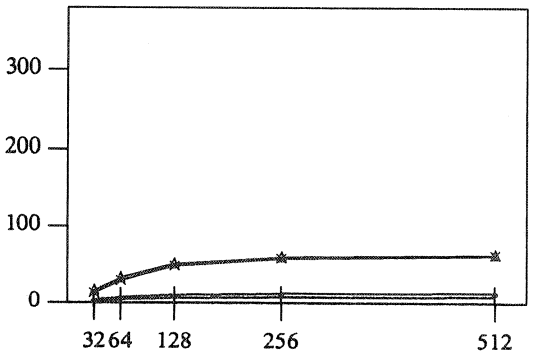

Fig.42g, NEC : Timing data for DSBTRS, Lower, κ=31

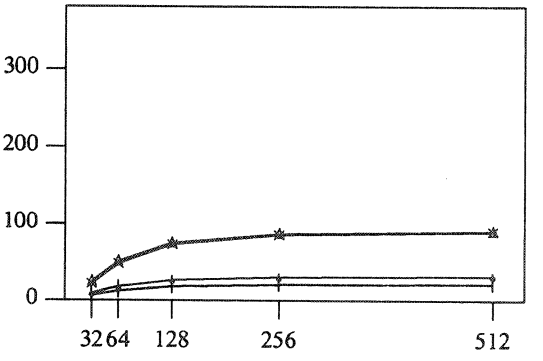

Fig.42h, NEC : Timing data for ZSBTRS, Lower, κ=31

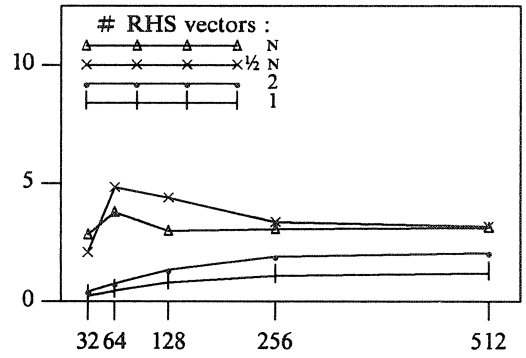

Fig.43a, FX4 : Timing data for DSBTRS, Lower, K=63

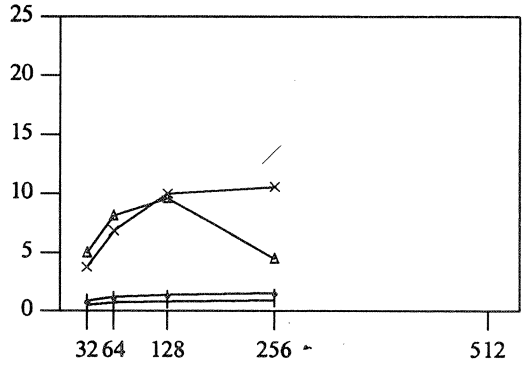

Fig.43b, 995 : Timing data for SSBTRS, Lower, K=63

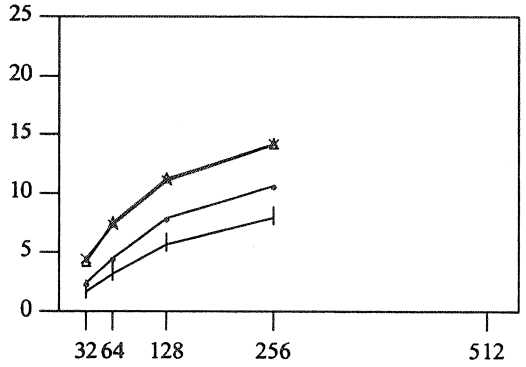

Fig.43c, IBM : Timing data for DSBTRS, Lower, K=63

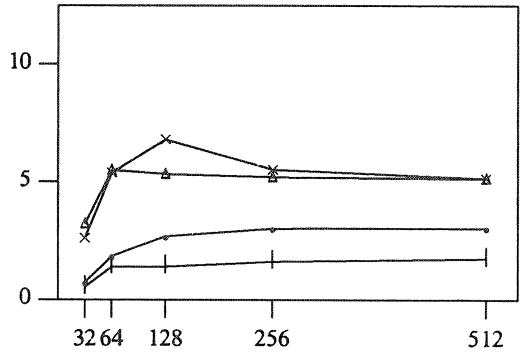

Fig.43d, FX4 : Timing data for ZSBTRS, Lower, K=63

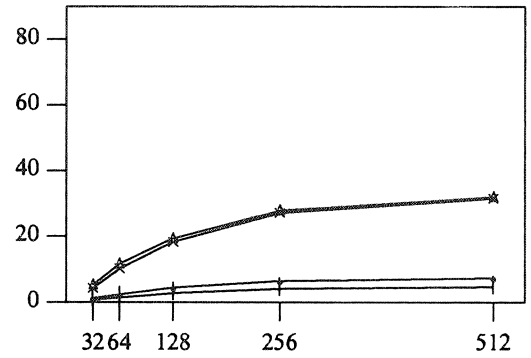

Fig.43e, 205 : Timing data for SSBTRS, Lower, K=63

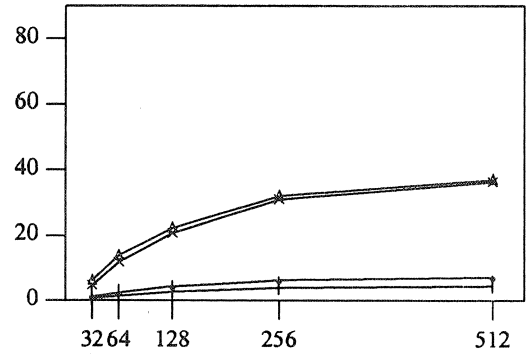

Fig.43f, 205.opt : Timing data for SSBTRS, Lower, K=63

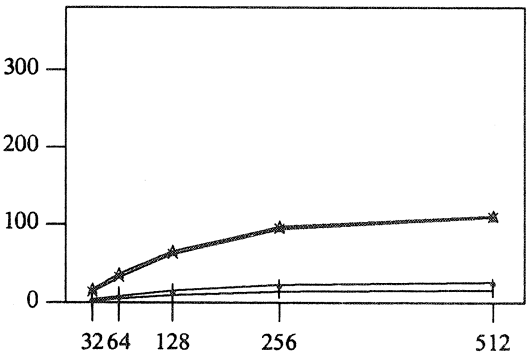

Fig.43g, NEC : Timing data for DSBTRS, Lower, K=63

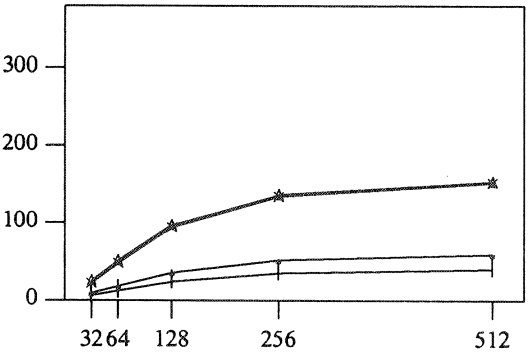

Fig.43h, NEC : Timing data for ZSBTRS, Lower, K=63

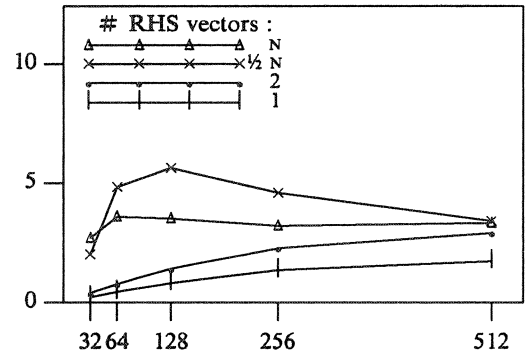

Fig.44a, FX4 : Timing data for DSBTRS, Lower, κ = 127

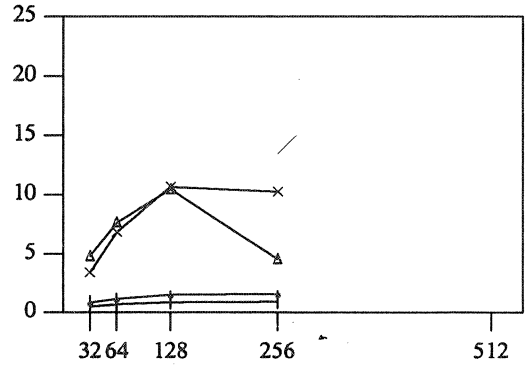

Fig.44b, 995 : Timing data for SSBTRS, Lower, κ = 127

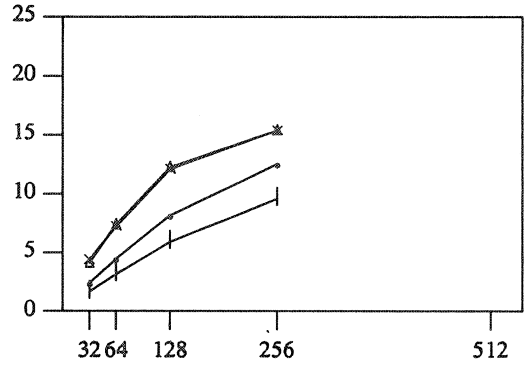

Fig.44c, IBM : Timing data for DSBTRS, Lower, κ = 127

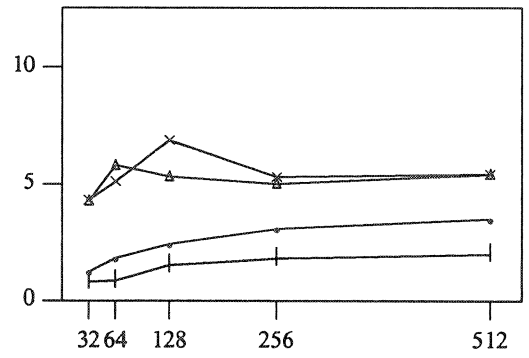

Fig.44d, FX4 : Timing data for ZSBTRS, Lower, κ = 127

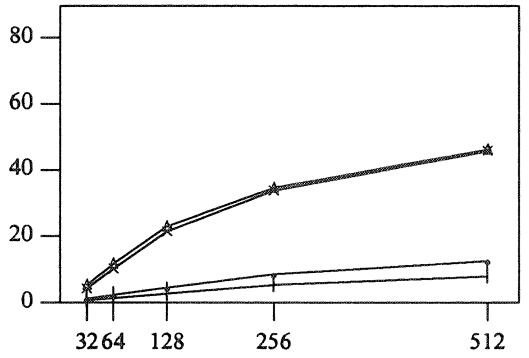

Fig.44e, 205 : Timing data for SSBTRS, Lower, κ = 127

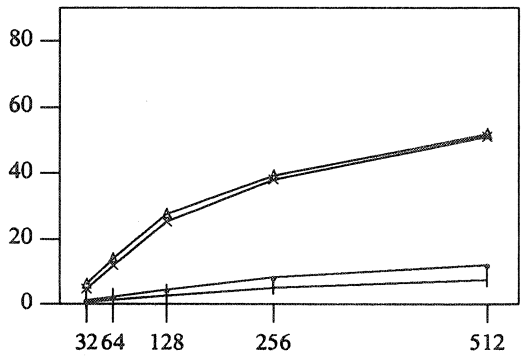

Fig.44f, 205.opt : Timing data for SSBTRS, Lower, κ = 127

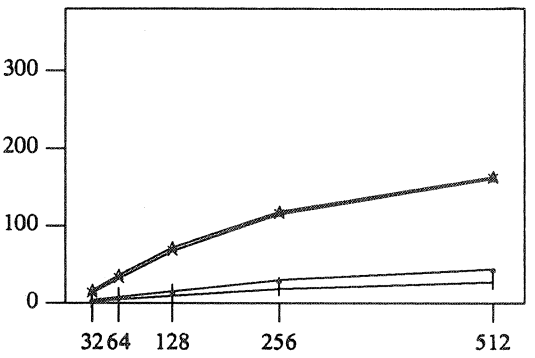

Fig.44g, NEC : Timing data for DSBTRS, Lower, κ = 127

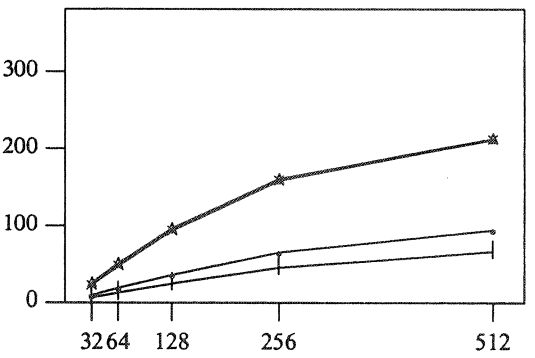

Fig.44h, NEC : Timing data for ZSBTRS, Lower, κ = 127

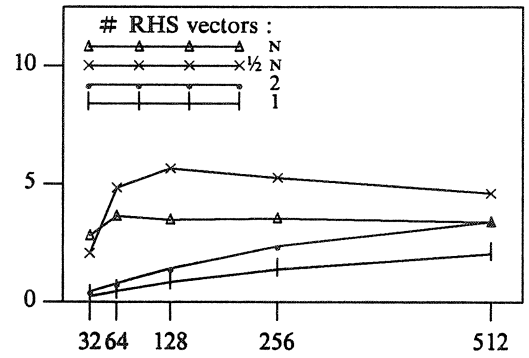

Fig.45a, FX4 : Timing data for DSBTRS, Lower, K=255

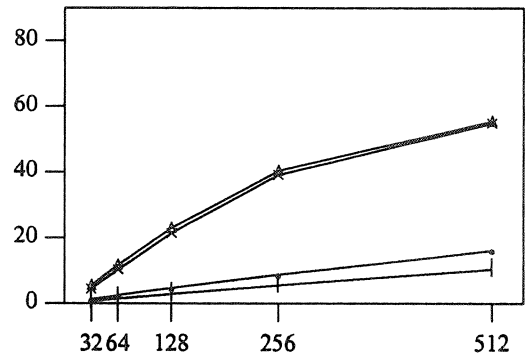

Fig.45e, 205 : Timing data for SSBTRS, Lower, K=255

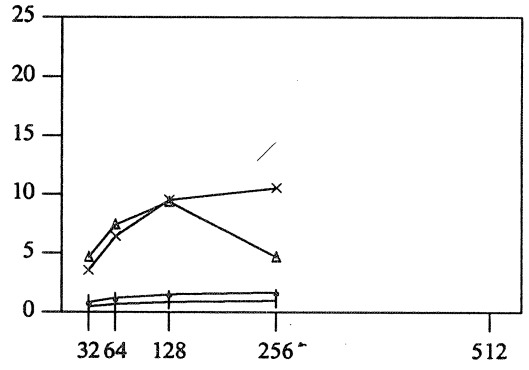

Fig.45b, 995 : Timing data for SSBTRS, Lower, K=255

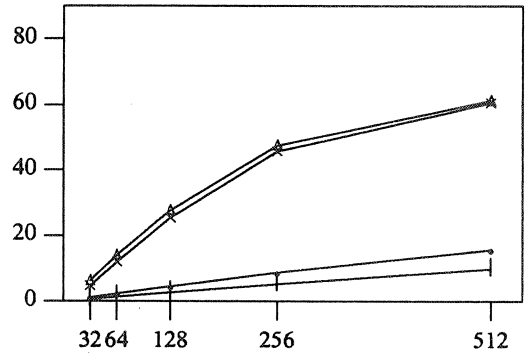

Fig.45f, 205.opt : Timing data for SSBTRS, Lower, K=255

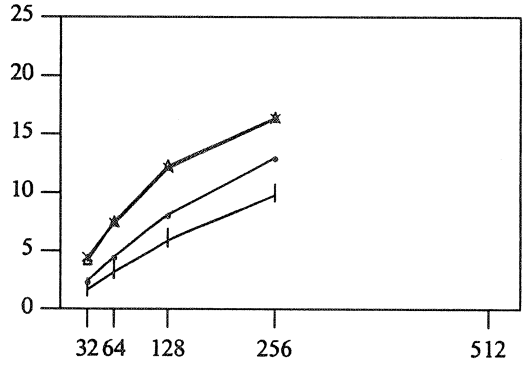

Fig.45c, IBM : Timing data for DSBTRS, Lower, K=255

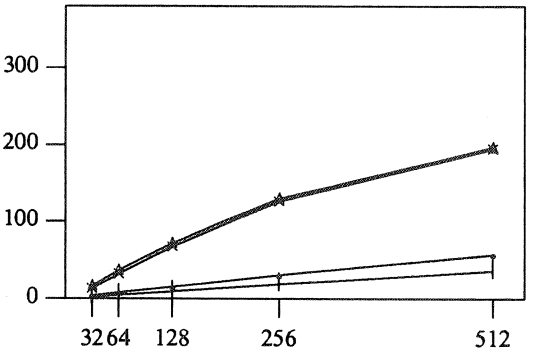

Fig.45g, NEC : Timing data for DSBTRS, Lower, K=255

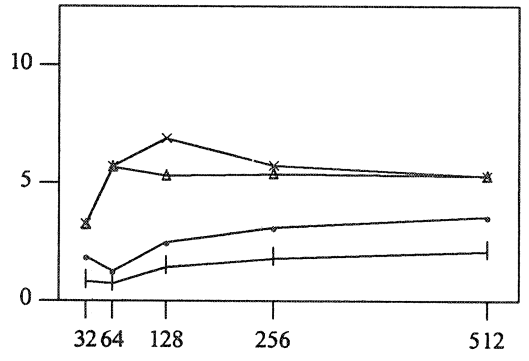

Fig.45d, FX4 : Timing data for ZSBTRS, Lower, K=255

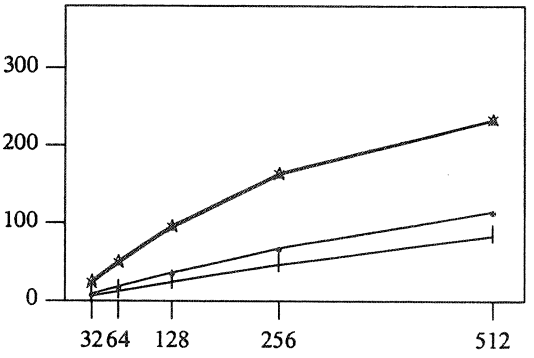

Fig.45h, NEC : Timing data for ZSBTRS, Lower, K=255

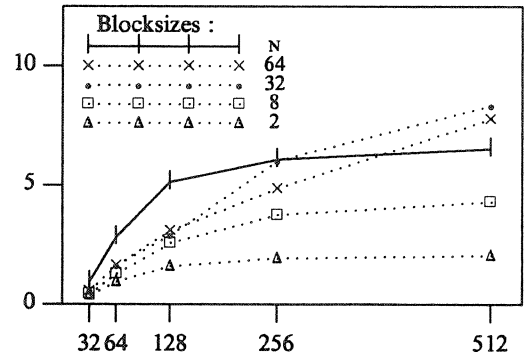

Fig.46a, FX4 : Timing data for DGEQRF, N = 1*M/3

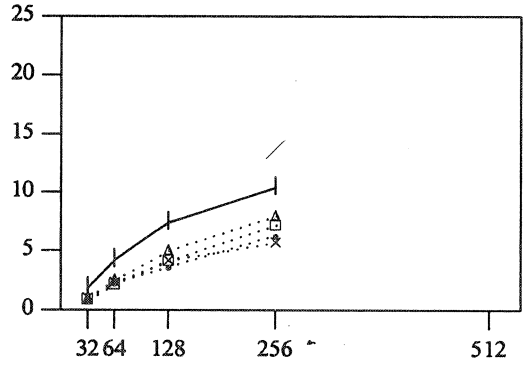

Fig.46b, 995 : Timing data for SGEQRF, N = 1*M/3

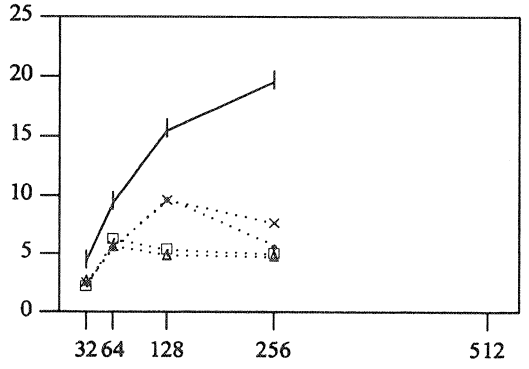

Fig.46c, IBM : Timing data for DGEQRF, N = 1*M/3

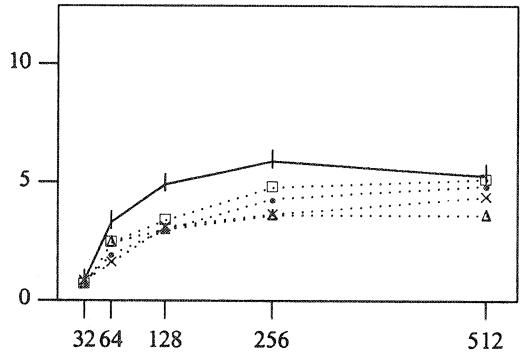

Fig.46d, FX4 : Timing data for ZGEQRF, N = 1*M/3

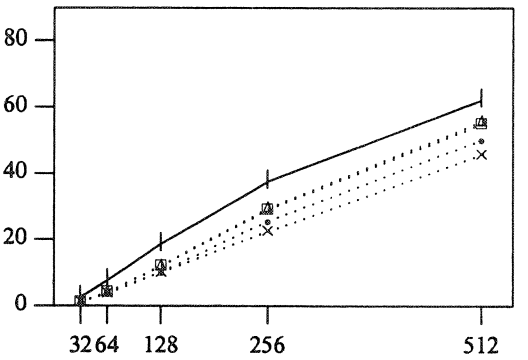

Fig.46e, 205 : Timing data for SGEQRF, N = 1*M/3

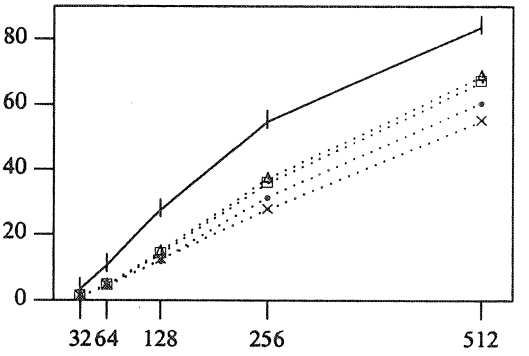

Fig.46f, 205.opt : Timing data for SGEQRF, N = 1*M/3

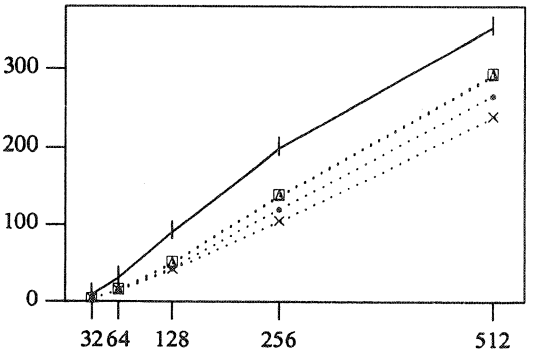

Fig.46g, NEC : Timing data for DGEQRF, N = 1*M/3

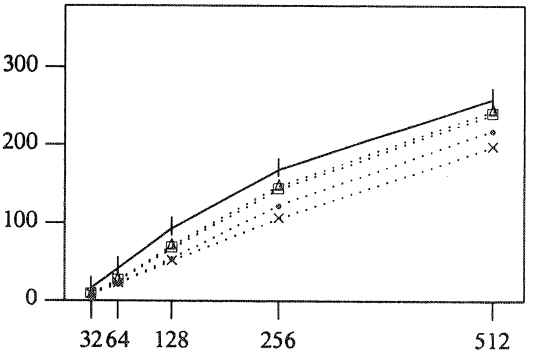

Fig.46h, NEC : Timing data for ZGEQRF, N = 1*M/3

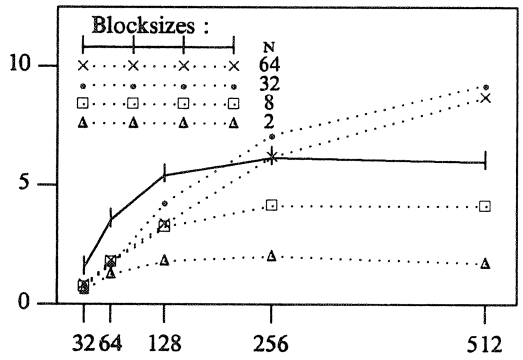

Fig.47a, FX4 : Timing data for DGEQRF, N=2*M/3

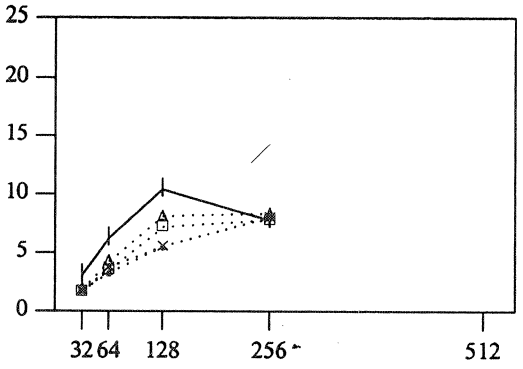

Fig.47b, 995 : Timing data for SGEQRF, N=2*M/3

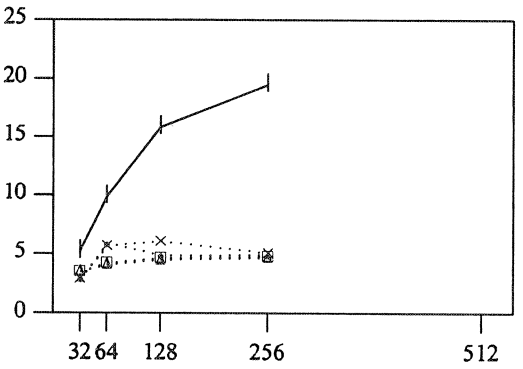

Fig.47c, IBM : Timing data for DGEQRF, N=2*M/3

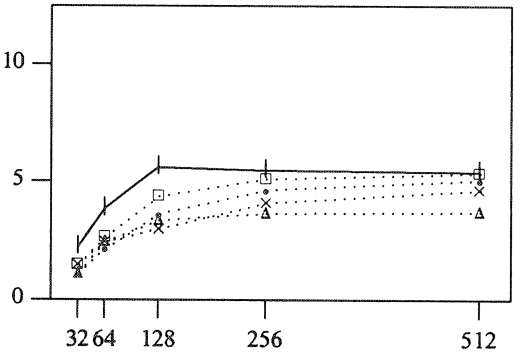

Fig.47d, FX4 : Timing data for ZGEQRF, N = 2*M/3

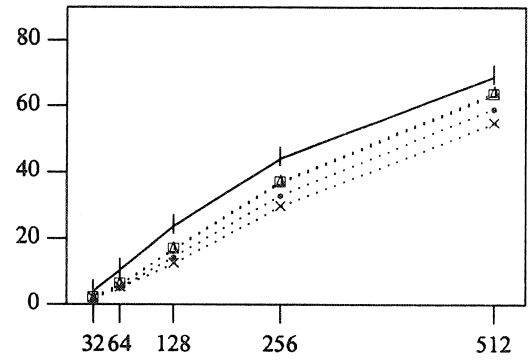

Fig.47e, 205 : Timing data for SGEQRF, N = 2*M/3

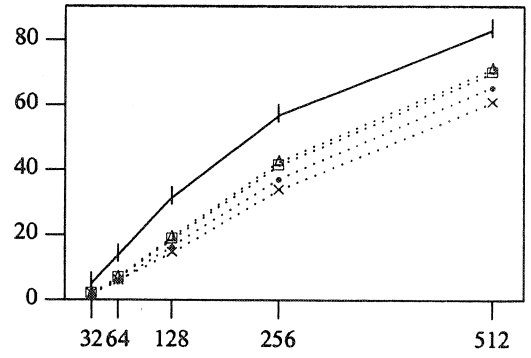

Fig.47f, 205.opt : Timing data for SGEQRF, N = 2*M/3

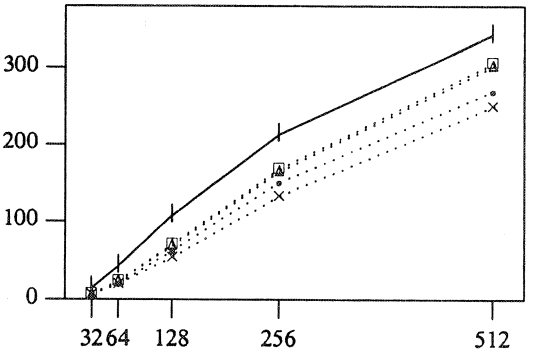

Fig.47g, NEC : Timing data for DGEQRF, N = 2*M/3

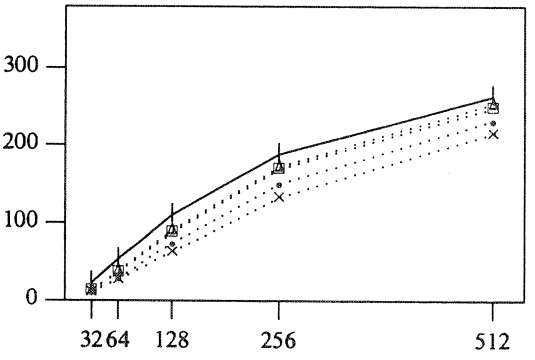

Fig.47h, NEC : Timing data for ZGEQRF, N = 2*M/3

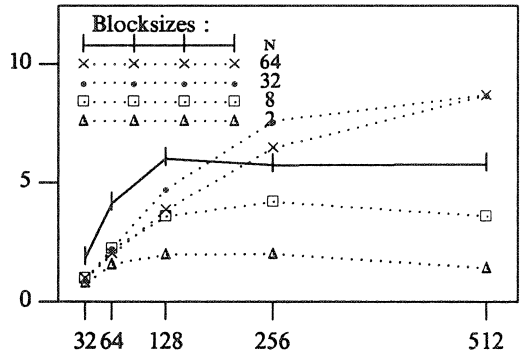

Fig.48a, FX4 : Timing data for DGEQRF, N = M

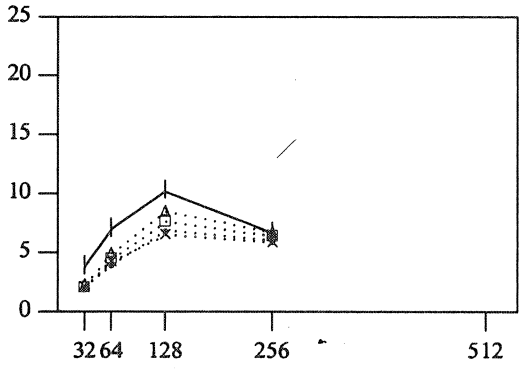

Fig.48b, 995 : Timing data for SGEQRF, N = M

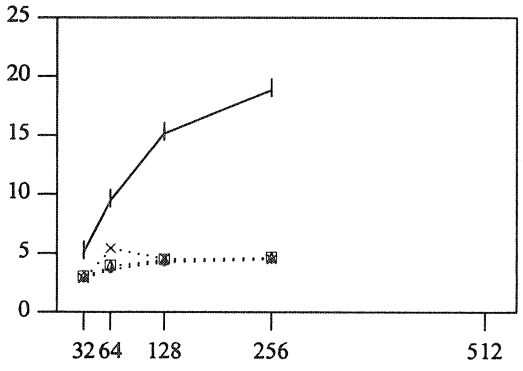

Fig.48c, IBM : Timing data for DGEQRF, N = M

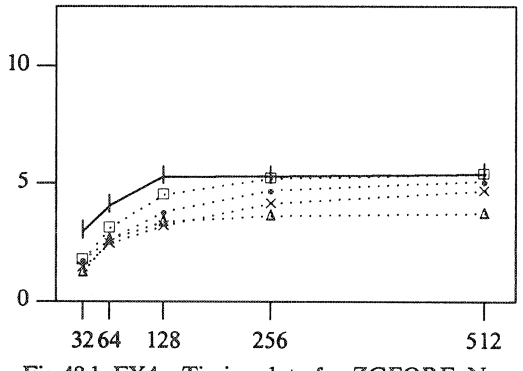

Fig.48d, FX4 : Timing data for ZGEQRF, N = M

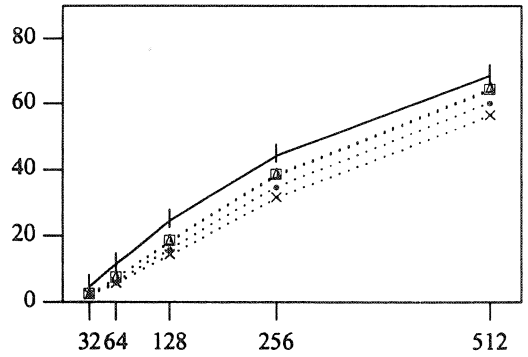

Fig.48e, 205 : Timing data for SGEQRF, N = M

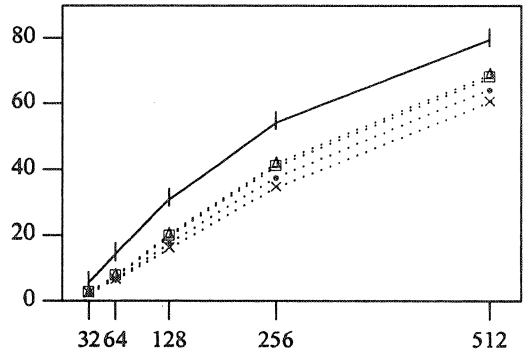

Fig.48f, 205.opt : Timing data for SGEQRF, N = M

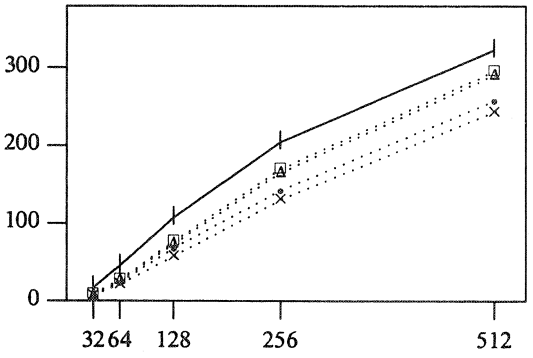

Fig.48g, NEC : Timing data for DGEQRF, N = M

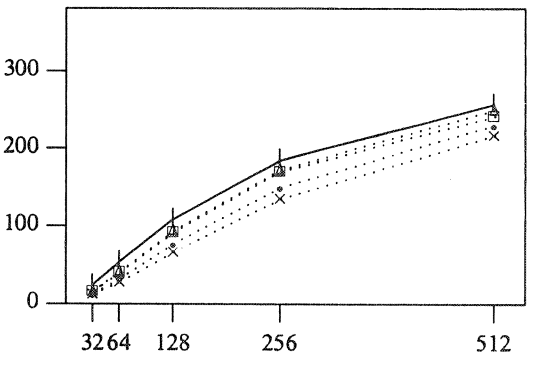

Fig.48h, NEC : Timing data for ZGEQRF, N = M

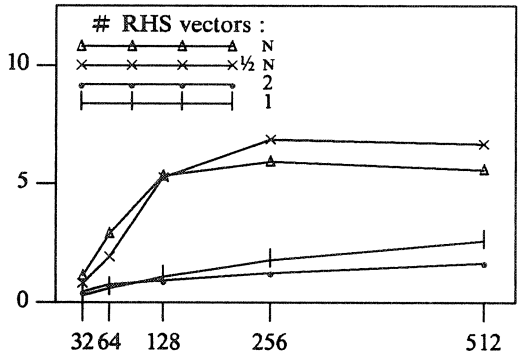

Fig.49a, FX4 : Timing data for DGEQRS, N = 1*M/3

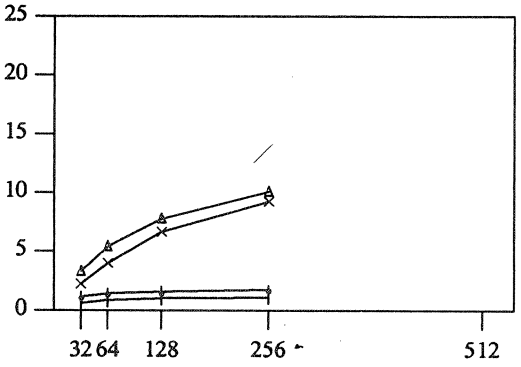

Fig.49b, 995 : Timing data for SGEQRS, N = 1*M/3

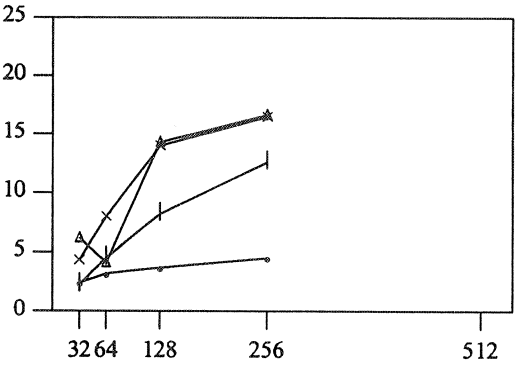

Fig.49c, IBM : Timing data for DGEQRS, N = 1*M/3

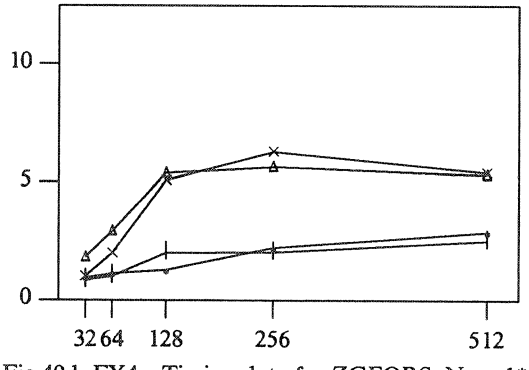

Fig.49d, FX4 : Timing data for ZGEQRS, N = 1*M/3

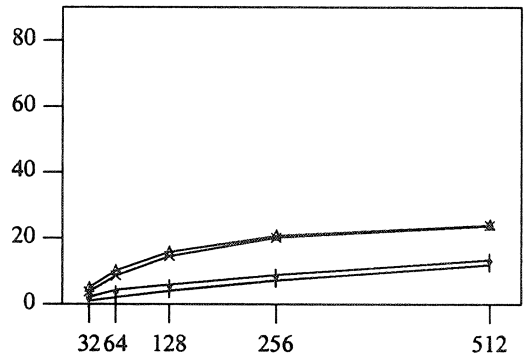

Fig.49e, 205 : Timing data for SGEQRS, N = 1*M/3

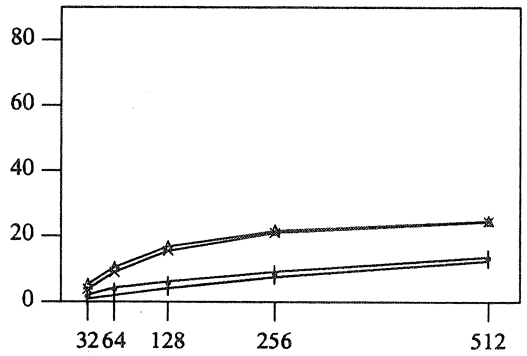

Fig.49f, 205.opt : Timing data for SGEQRS, N = 1*M/3

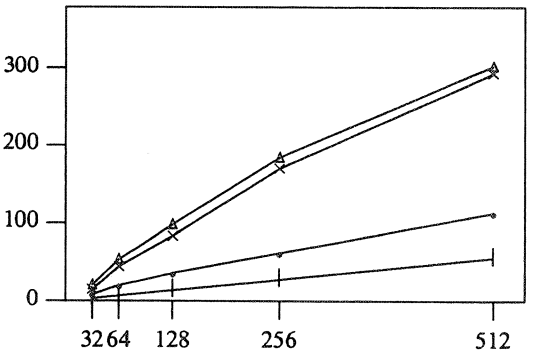

Fig.49g, NEC : Timing data for DGEQRS, N = 1*M/3

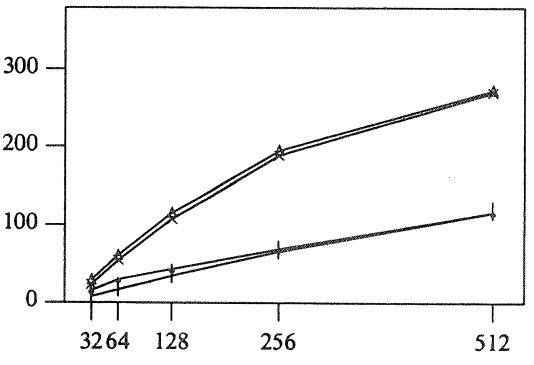

Fig.49h, NEC : Timing data for ZGEQRS, N = 1*M/3

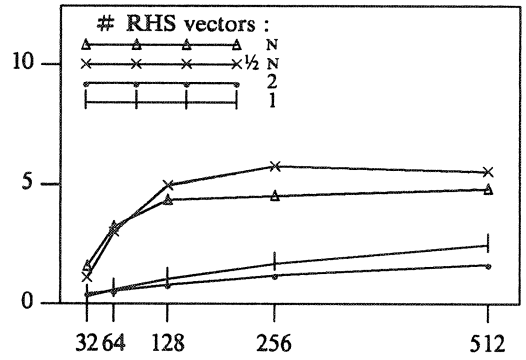

Fig.50a, FX4 : Timing data for DGEQRS, N=2*M/3

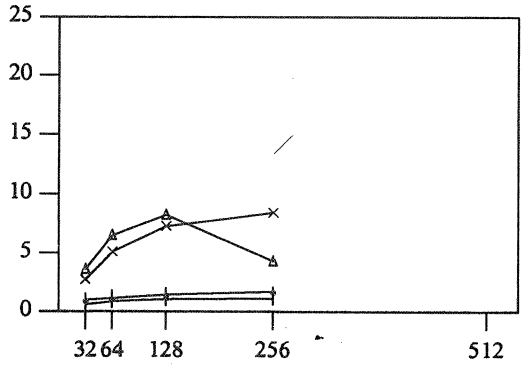

Fig.50b, 995 : Timing data for SGEQRS, N=2*M/3

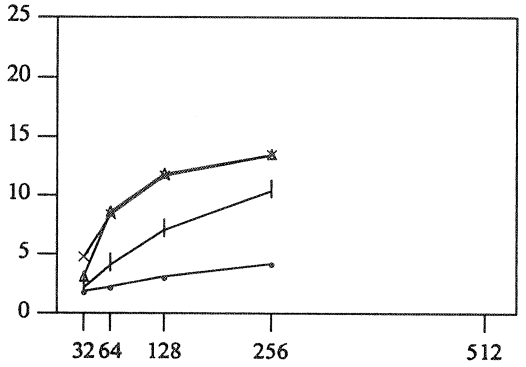

Fig.50c, IBM : Timing data for DGEQRS, N=2*M/3

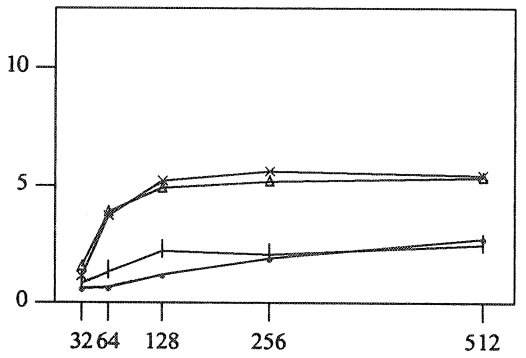

Fig.50d, FX4 : Timing data for ZGEQRS, N = 2*M/3

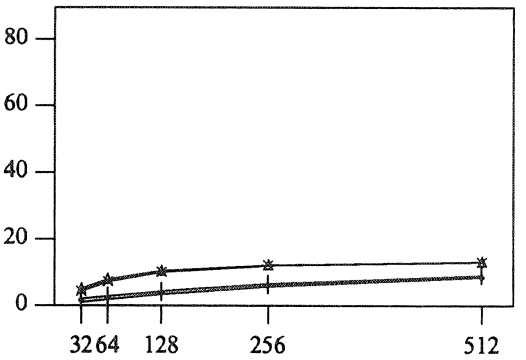

Fig.50e, 205 : Timing data for SGEQRS, N = 2*M/3

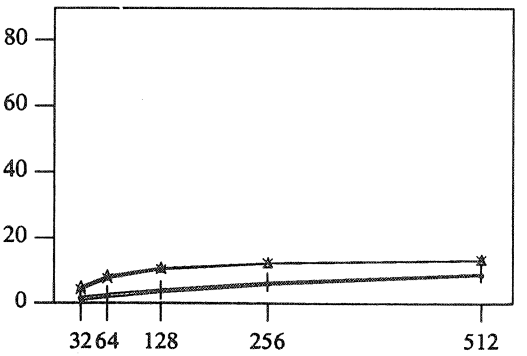

Fig.50f, 205.opt : Timing data for SGEQRS, N = 2*M/3

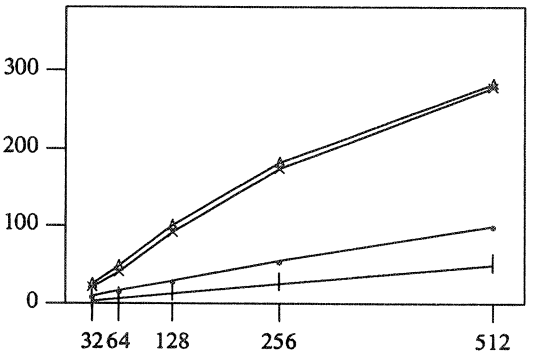

Fig.50g, NEC : Timing data for DGEQRS, N = 2*M/3

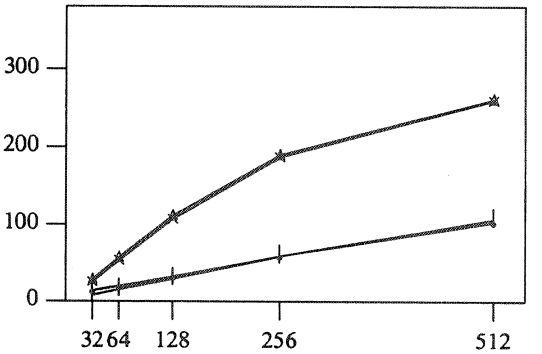

Fig.50h, NEC : Timing data for ZGEQRS, N = 2*M/3

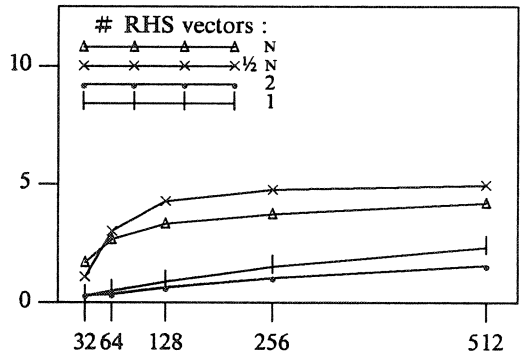

Fig.51a, FX4 : Timing data for DGEQRS, N = M

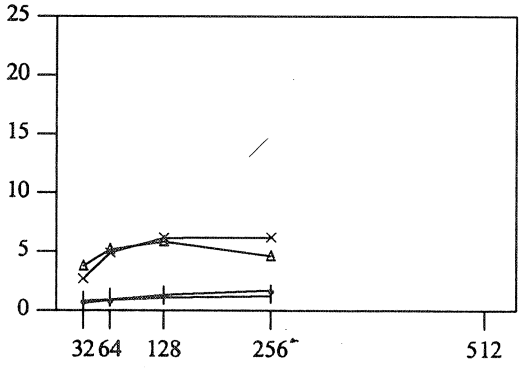

Fig.51b, 995 : Timing data for SGEQRS, N = M

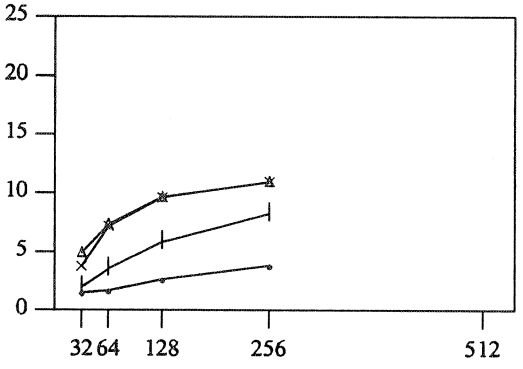

Fig.51c, IBM : Timing data for DGEQRS, N = M

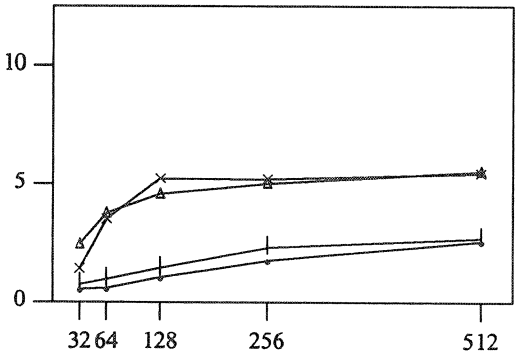

Fig.51d, FX4 : Timing data for ZGEQRS, N = M

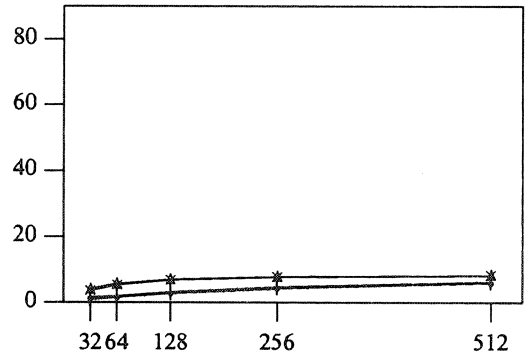

Fig.51e, 205 : Timing data for SGEQRS, N = M

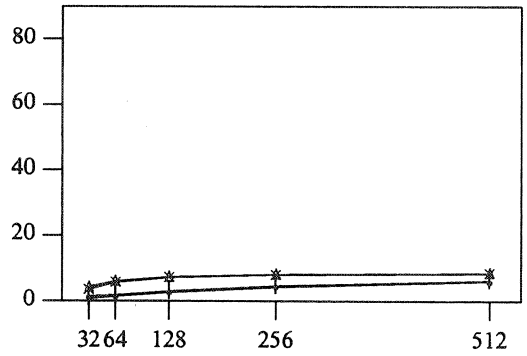

Fig.51f, 205.opt : Timing data for SGEQRS, N = M

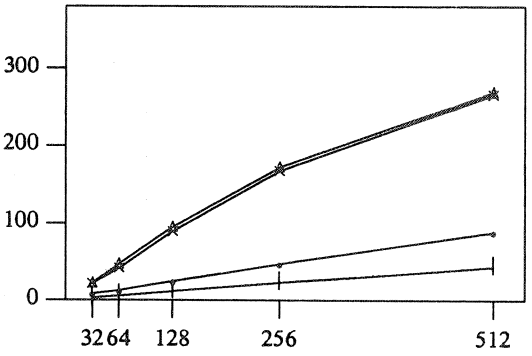

Fig.51g, NEC : Timing data for DGEQRS, N = M

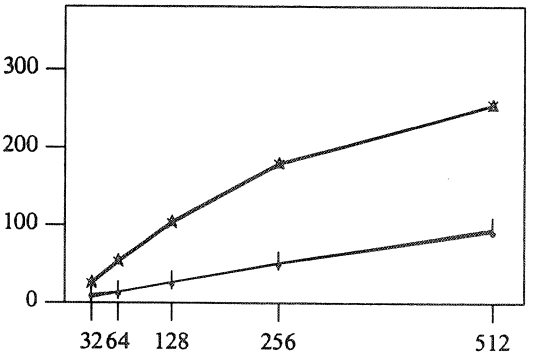

Fig.51h, NEC : Timing data for ZGEQRS, N = M

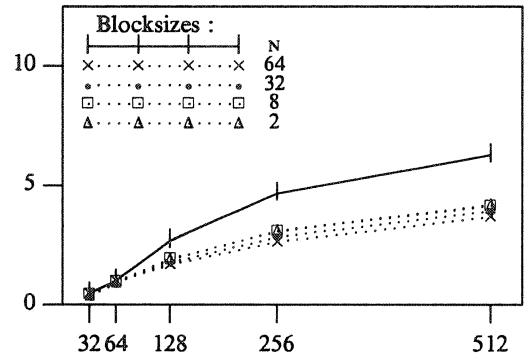

Fig.52a, FX4 : Timing data for DTRTRI, Upper

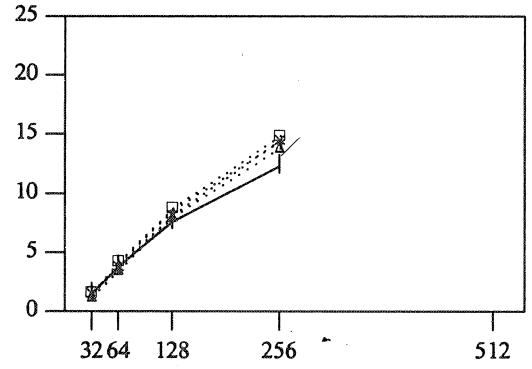

Fig.52b, 995 : Timing data for STRTRI, Upper

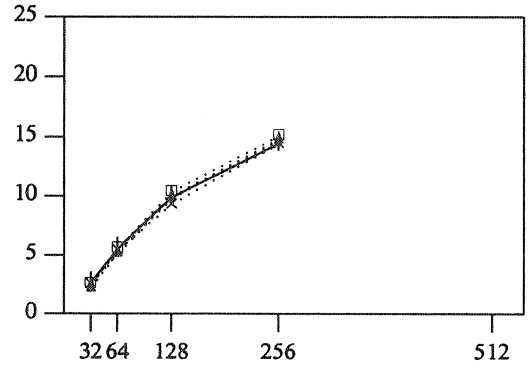

Fig.52c, IBM : Timing data for DTRTRI, Upper

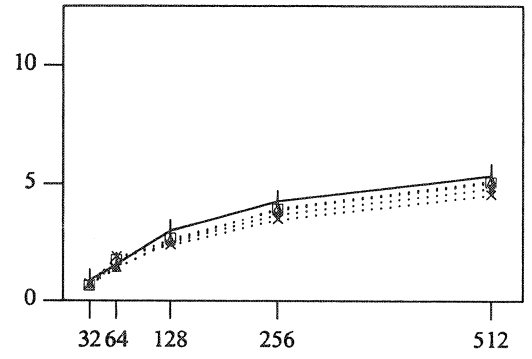

Fig.52d, FX4 : Timing data for ZTRTRI, Upper

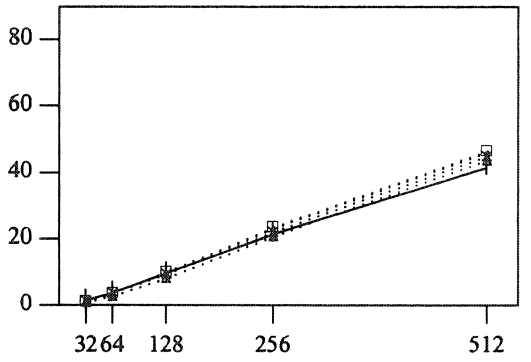

Fig.52e, 205 : Timing data for STRTRI, Upper

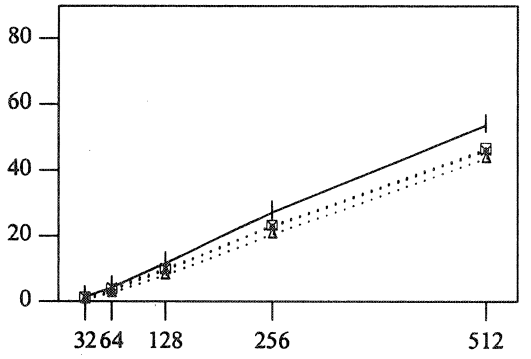

Fig.52f, 205.opt : Timing data for STRTRI, Upper

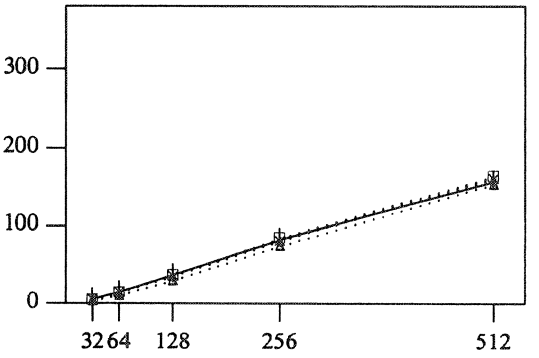

Fig.52g, NEC : Timing data for DTRTRI, Upper

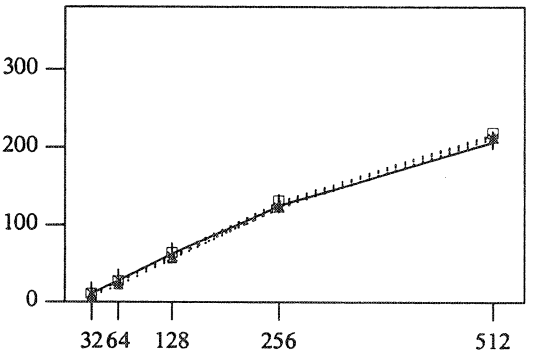

Fig.52h, NEC : Timing data for ZTRTRI, Upper

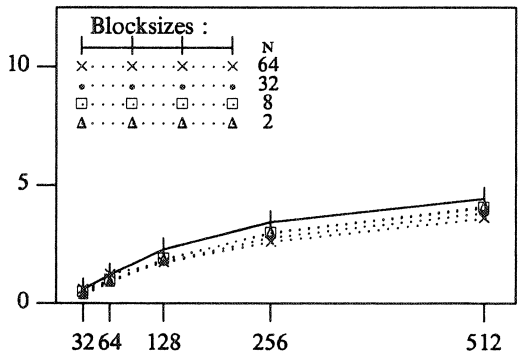

Fig.53a, FX4 : Timing data for DTRTRI, Lower

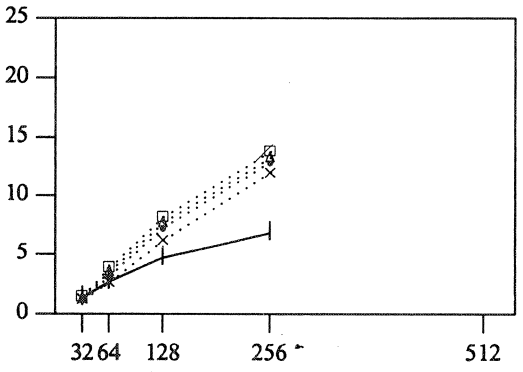

Fig.53b, 995 : Timing data for STRTRI, Lower

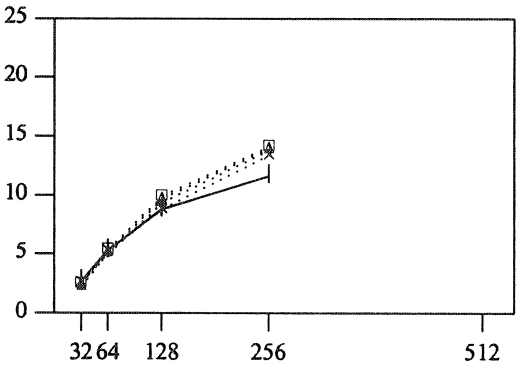

Fig.53c, IBM : Timing data for DTRTRI, Lower

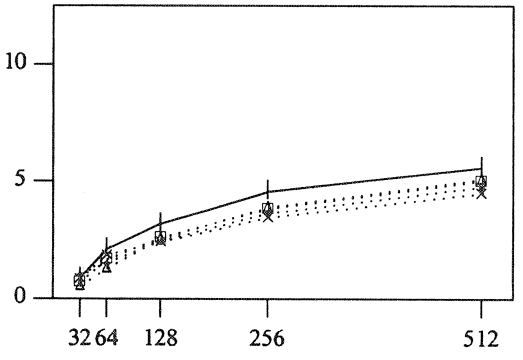

Fig.53d, FX4 : Timing data for ZTRTRI, Lower

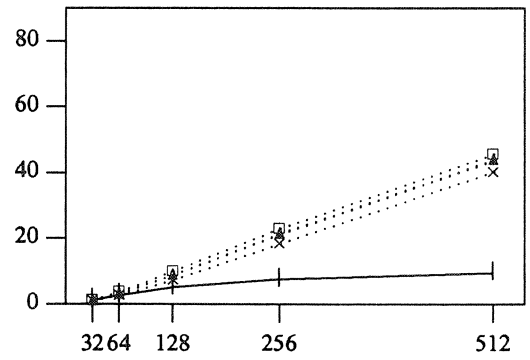

Fig.53e, 205 : Timing data for STRTRI, Lower

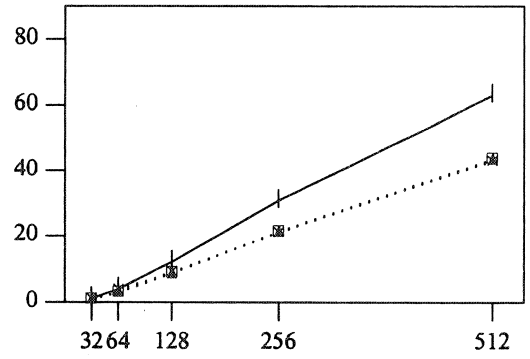

Fig.53f, 205.opt : Timing data for STRTRI, Lower

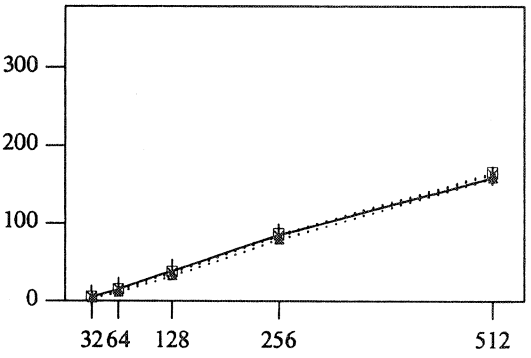

Fig.53g, NEC : Timing data for DTRTRI, Lower

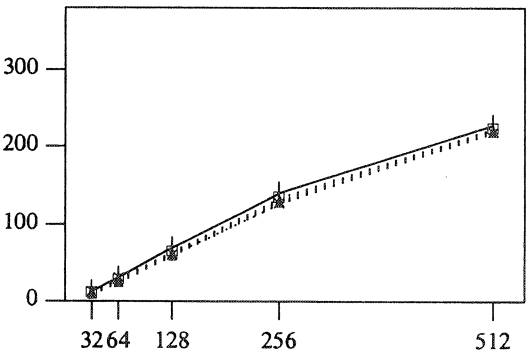

Fig.53h, NEC : Timing data for ZTRTRI, Lower

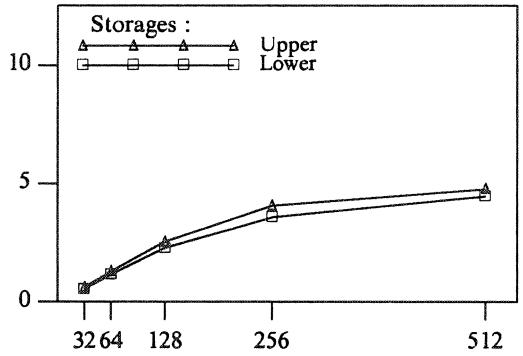

Fig.54a, FX4 : Timing data for DTPTRI

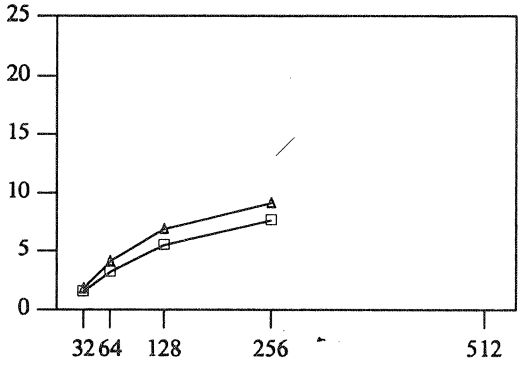

Fig.54b, 995 : Timing data for STPTRI

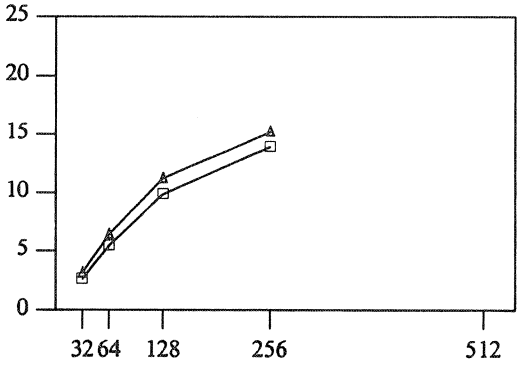

Fig.54c, IBM : Timing data for DTPTRI

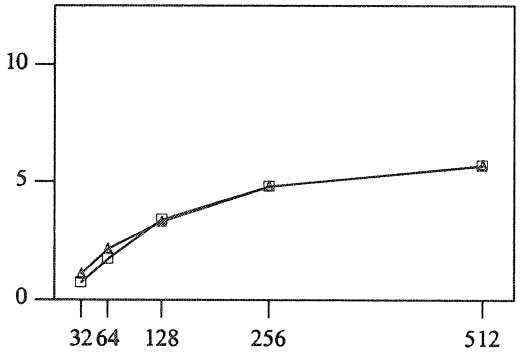

Fig.54d, FX4 : Timing data for ZTPTRI

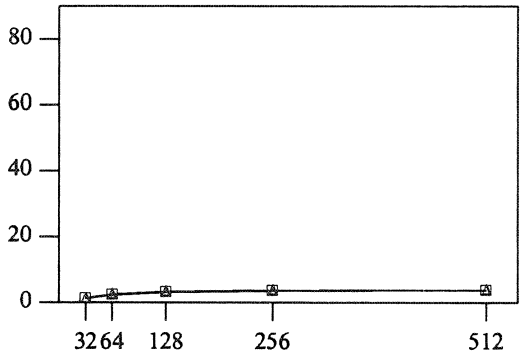

Fig.54e, 205 : Timing data for STPTRI

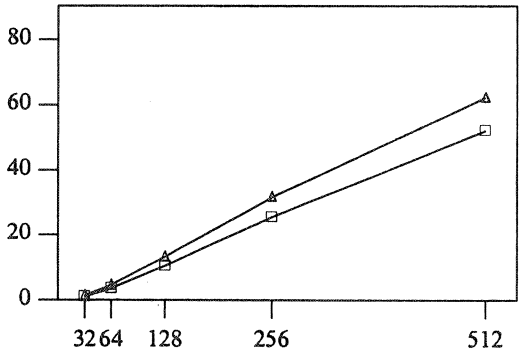

Fig.54f, 205.opt : Timing data for STPTRI

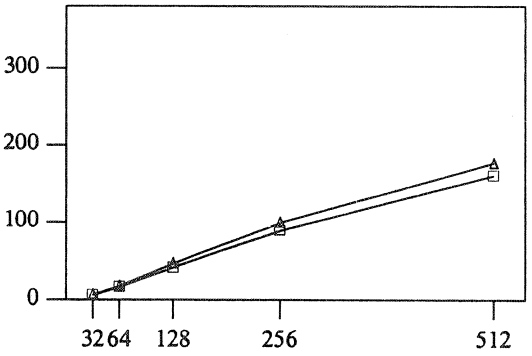

Fig.54g, NEC : Timing data for DTPTRI

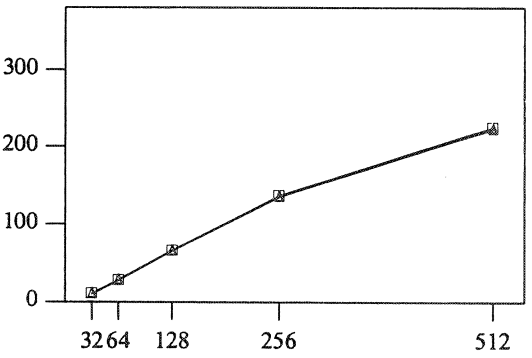

Fig.54h, NEC : Timing data for ZTPTRI

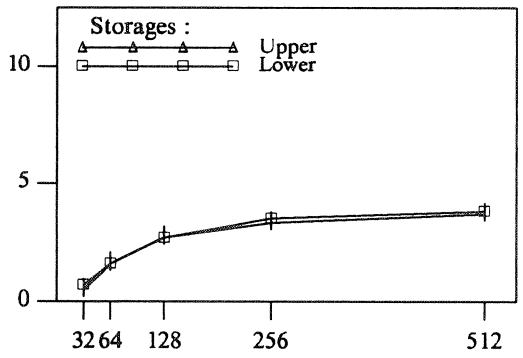

Fig.55d, FX4 : Timing data for ZHETRF

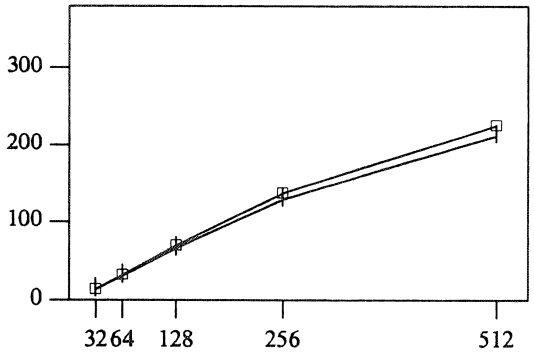

Fig.55h, NEC : Timing data for ZHETRF

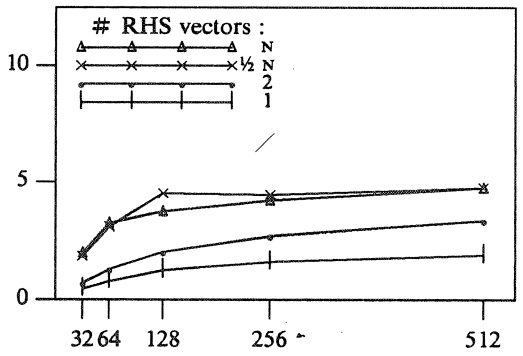

Fig.56d, FX4 : Timing data for ZHETRS, Upper

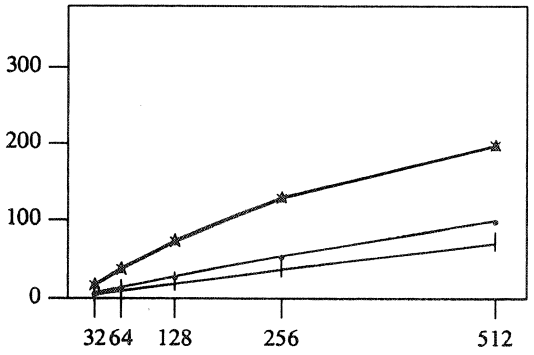

Fig.56h, NEC : Timing data for ZHETRS, Upper

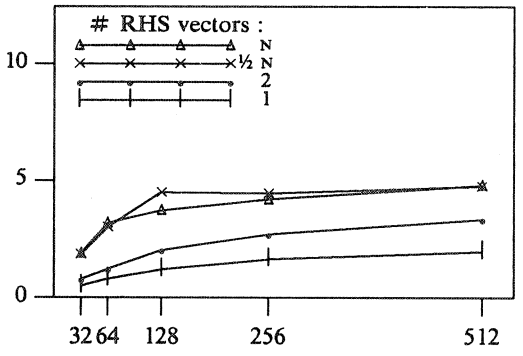

Fig.57d, FX4 : Timing data for ZHETRS, Lower

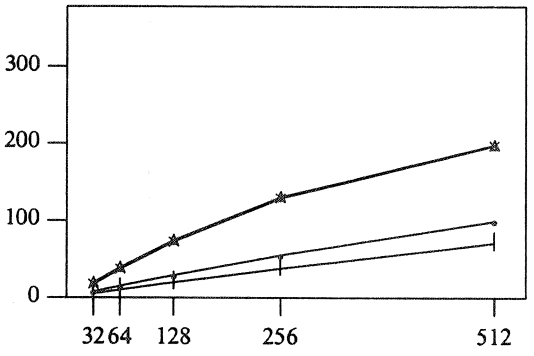

Fig.57h, NEC : Timing data for ZHETRS, Lower

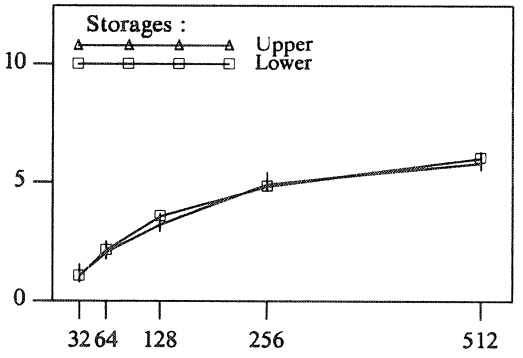

Fig.58d, FX4 : Timing data for ZHETRI

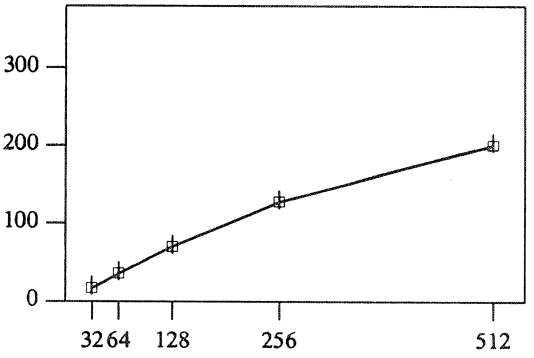

Fig.58h, NEC : Timing data for ZHETRI

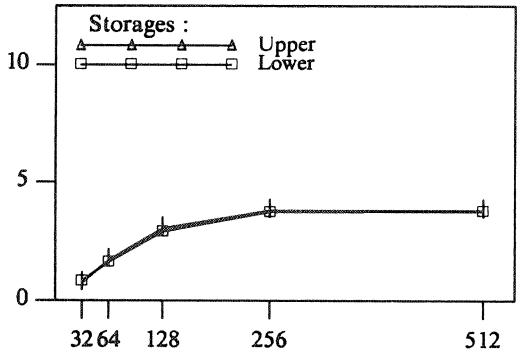

Fig.59d, FX4 : Timing data for ZHPTRF

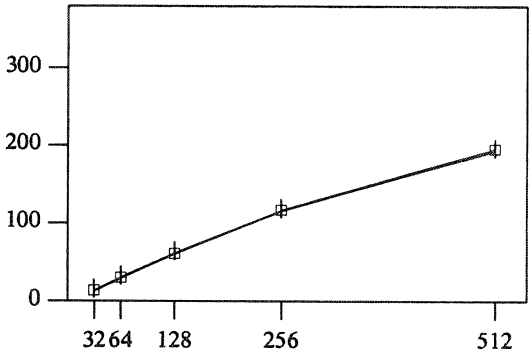

Fig.59h, NEC : Timing data for ZHPTRF

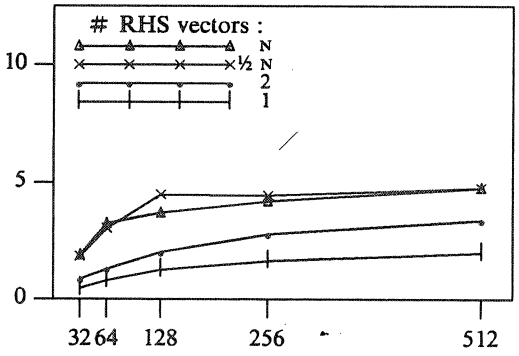

Fig.60d, FX4 : Timing data for ZHPTRS, Upper

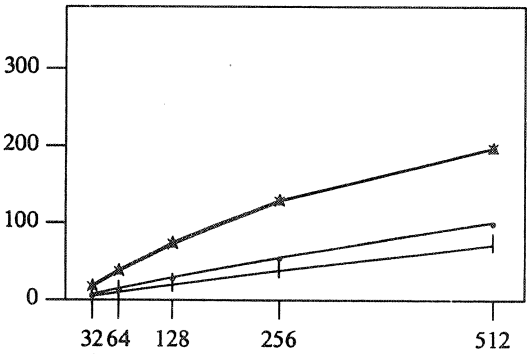

Fig.60h, NEC : Timing data for ZHPTRS, Upper

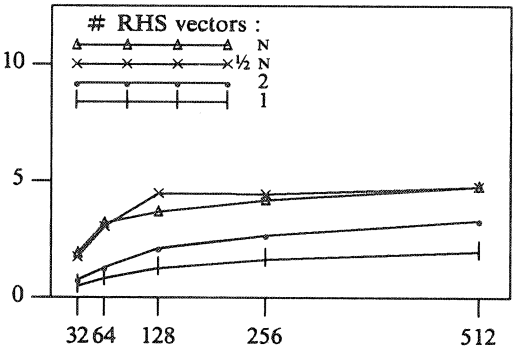

Fig.61d, FX4 : Timing data for ZHPTRS, Lower

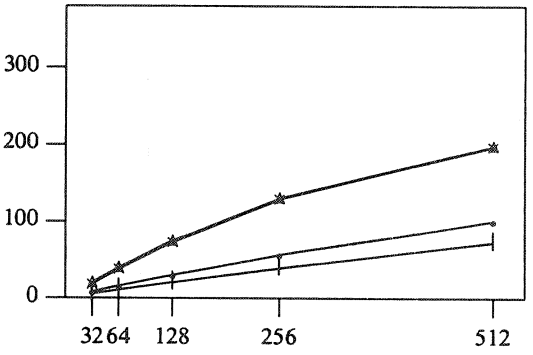

Fig.61h, NEC : Timing data for ZHPTRS, Lower

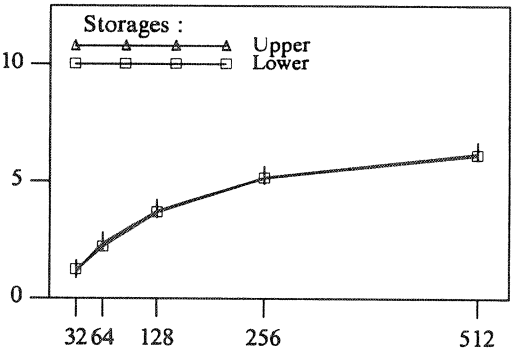

Fig.62d, FX4 : Timing data for ZHPTRI

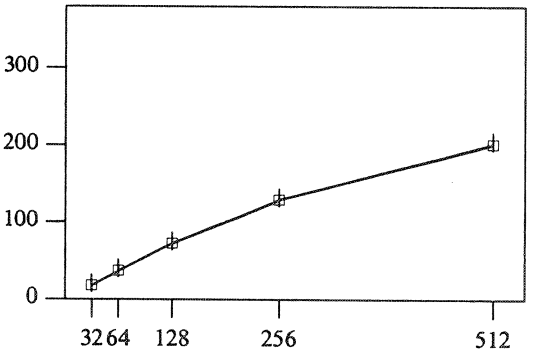

Fig.62h, NEC : Timing data for ZHPTRI

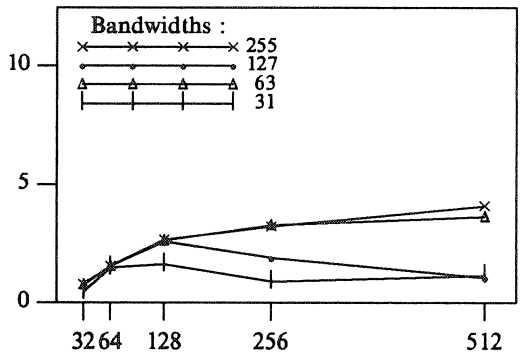

Fig.63d, FX4 : Timing data for ZHBTRF, Upper

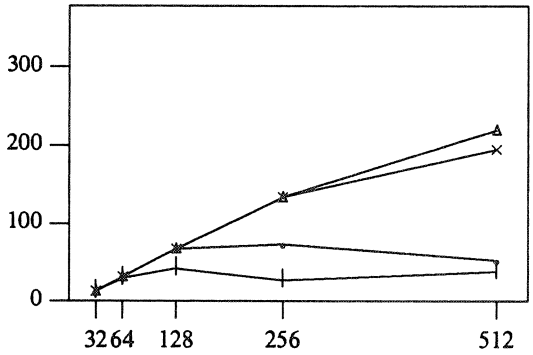

Fig.63h, NEC : Timing data for ZHBTRF, Upper

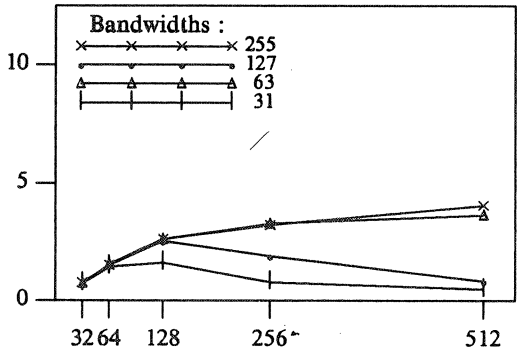

Fig.64d, FX4 : Timing data for ZHBTRF, Lower

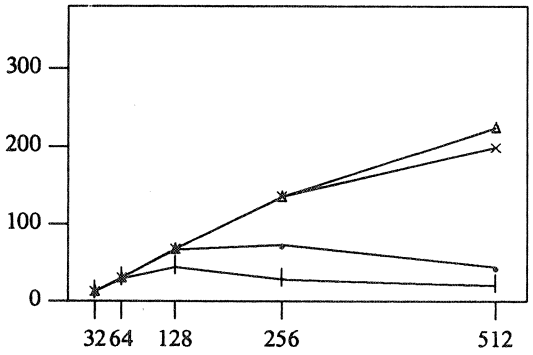

Fig.64h, NEC : Timing data for ZHBTRF, Lower

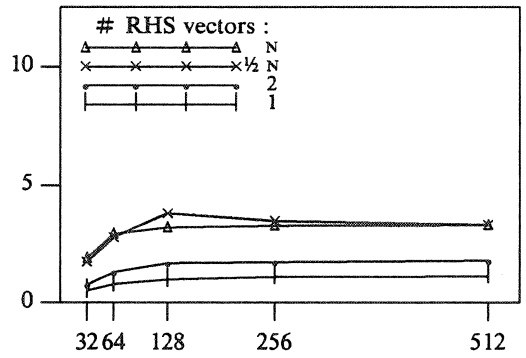

Fig.65d, FX4 : Timing data for ZHBTRS, Upper, K=31

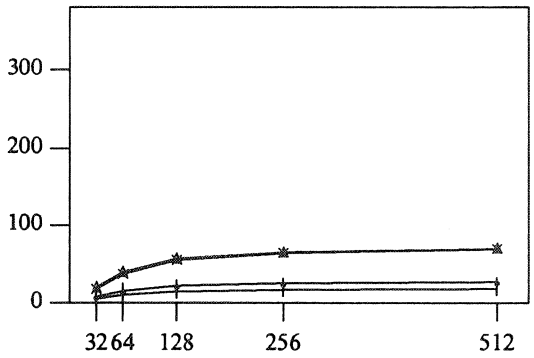

Fig.65h, NEC : Timing data for ZHBTRS, Upper, K=31

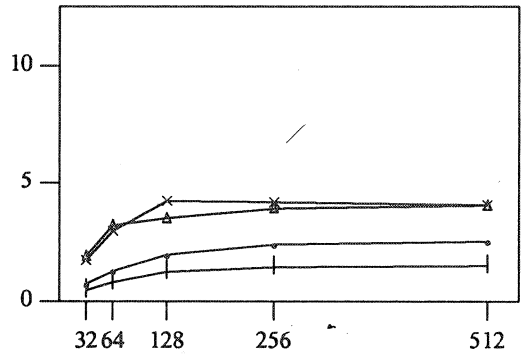

Fig.66d, FX4 : Timing data for ZHBTRS, Upper, K=63

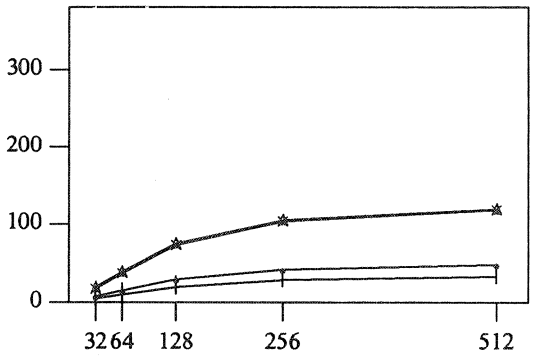

Fig.66h, NEC : Timing data for ZHBTRS, Upper, K=63

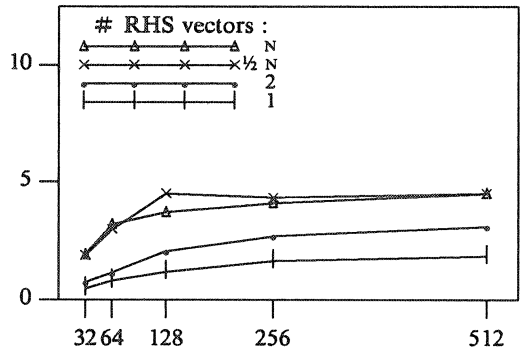

Fig.67d, FX4 : Timing data for ZHBTRS, Upper, K=127

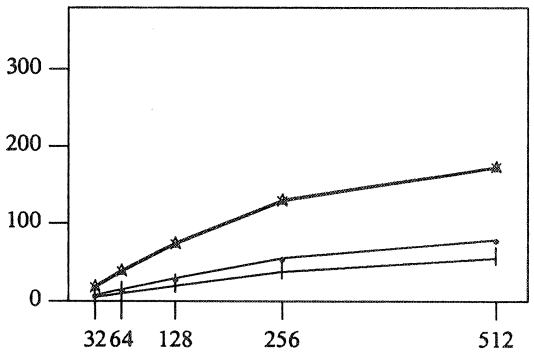

Fig.67h, NEC : Timing data for ZHBTRS, Upper, K=127

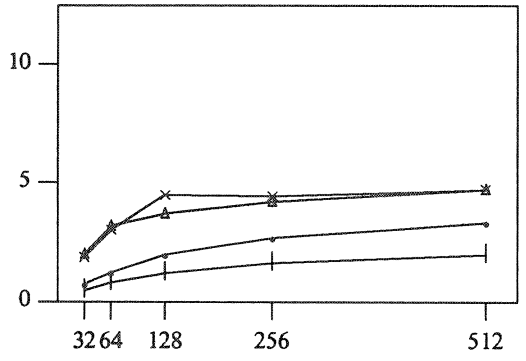

Fig.68d, FX4 : Timing data for ZHBTRS, Upper, K=255

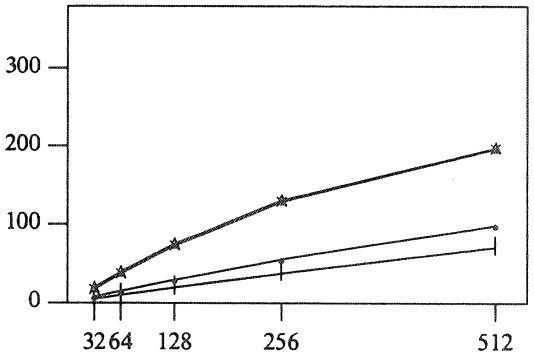

Fig.68h, NEC : Timing data for ZHBTRS, Upper, K=255

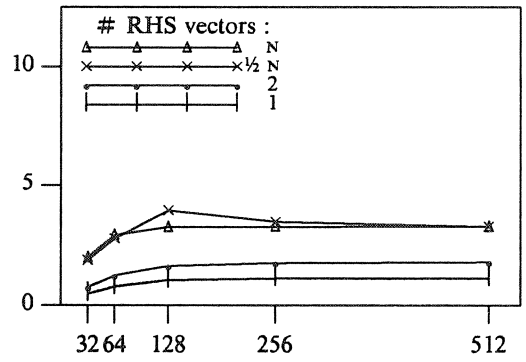

Fig.69d, FX4 : Timing data for ZHBTRS, Lower, K=31

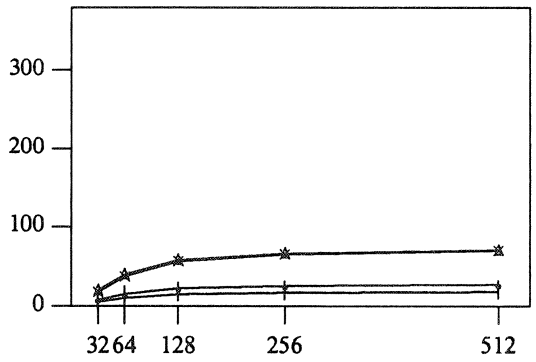

Fig.69h, NEC : Timing data for ZHBTRS, Lower, K=31

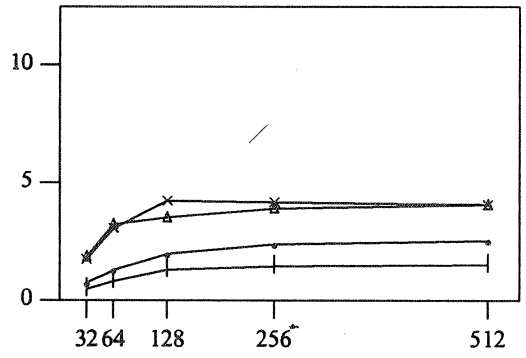

Fig.70d, FX4 : Timing data for ZHBTRS, Lower, K=63

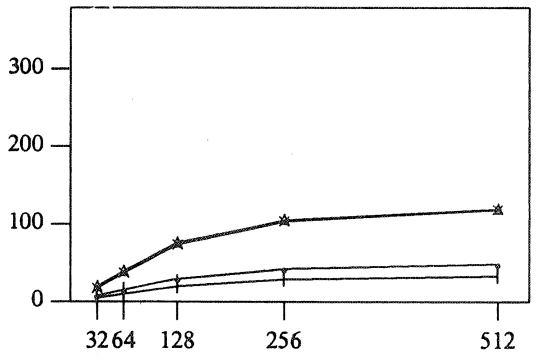

Fig.70h, NEC : Timing data for ZHBTRS, Lower, K=63

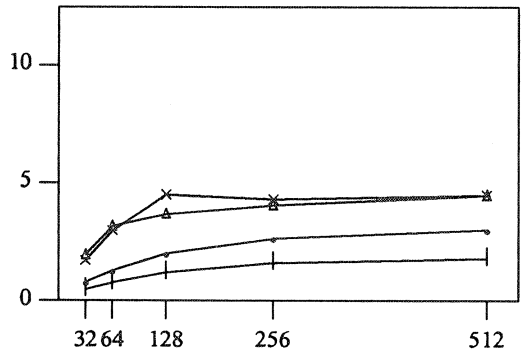

Fig.71d, FX4 : Timing data for ZHBTRS, Lower, K=127

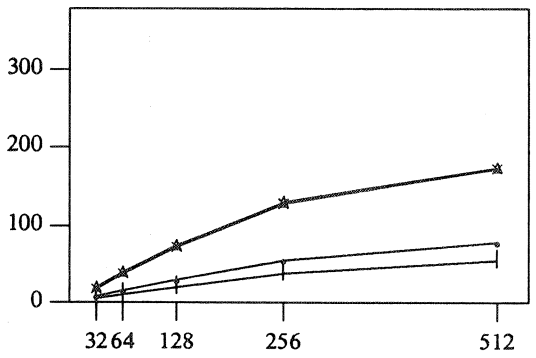

Fig.71h, NEC : Timing data for ZHBTRS, Lower, K=127

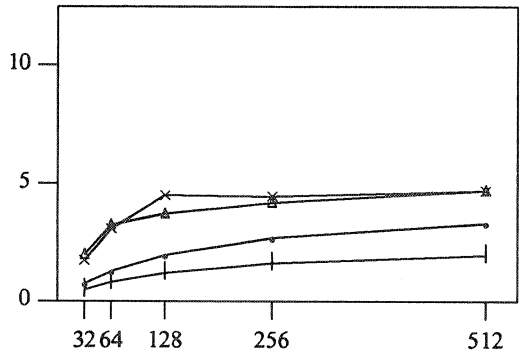

Fig.72d, FX4 : Timing data for ZHBTRS, Lower, K=255

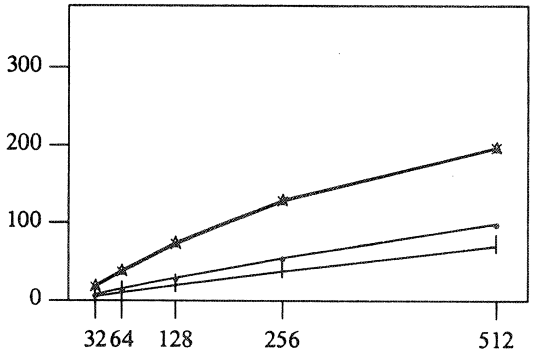

Fig.72h, NEC : Timing data for ZHBTRS, Lower, K=255