JOURNAL OF RESEARCH of the National Bureau of Standards—B. Mathematical Sciences
Vol. 78B, No. 1, January–March 1974

Comparison of Some FORTRAN Programs for Matrix Inversion*

Kenneth E. Fitzgerald**

(August 6, 1973)

In this paper several programs for computing the inverse of a matrix are compared primarily on the basis of execution time. Accuracy estimates and two programs that use iterative refinement are included. It is shown that for small matrices, improvement procedures are worthwhile but for large matrices, one must be more careful in their use. Two other points are also brought out: the value of multiplying matrices before taking the norm of a product and the need for some kind of an error estimate to be included in the output of every program.

Key words: Error estimates; evaluation of computer programs; execution time; inverse of a matrix; iterative refinement; linear systems.

Introduction

In a previous paper [1] a number of FORTRAN programs for finding the inverse of a matrix were compared solely on the basis of accuracy. Information with respect to the execution time of these programs is also of importance and we would like to discuss this factor in this paper. This latter element becomes a very important consideration when iterative refinement is used because a more accurate inverse will clearly result but we must evaluate the added time and effort it requires.

In this paper, then, we will briefly summarize the major results of the previous work in order to have them at hand, describe the programs and the test matrices to be used, present the information with respect to execution time as well as accuracy, and then discuss the results.

Review

Any discussion of accuracy involves the concept of a norm to determine the "size" of the error and we will use two: the Frobenius norm, where $N(A) = \sqrt{\Sigma\,|a_{ij}|^2}$, and the maximum element norm, where $N(A) = n \cdot \max_{ij} |a_{ij}|$. It is important to note the need for the multiplication by the size of the matrix in the latter case so that the second condition for a norm, $N(AB) \leq N(A)N(B)$, will be satisfied. With regard to this particular inequality, we would like to emphasize a point made in the previous paper. The derivation of many theoretical error bounds involves the use of this property and it is clearly much simpler and faster to calculate the norm of each matrix and multiply these two numbers than to multiply two matrices and then compute the norm. However, it has been our experience that the latter procedure gives a much smaller number than the former and as a result gives a much better indication of the value of the result. This can clearly be seen by looking at columns three and four in tables 1 to 10 that follow. The error bounds used in those tables were derived in the previous paper and we will just summarize the results now.

AMS Subject Classification: 65F05, 65F35.
*An invited paper.
**Present address: Manhattan College, New York, N.Y. 10471
[1] Figures in brackets indicate the literature references at the end of this paper.

Let X be an approximate inverse and let Y be the residual matrix $I-AX$. Then for any norm, if $N(Y)<1$, we have

1. $$\frac{N(X)}{1+N(Y)} \leqslant N(A^{-1}) \leqslant \frac{N(X)}{1-N(Y)}$$

2. (a) $$N(A^{-1}-X) \leqslant \frac{N(XY)}{1-N(Y)}$$

   (b) $$N(A^{-1}-X) \leqslant \frac{N(X)N(Y)}{1-N(Y)}$$

3. $$\frac{N(E)}{2N(A)} \leqslant N(A^{-1}-X),\ E=(I-XA)-(I-AX)$$

4. $$1-N(Y) \leqslant \frac{N(X)}{N(A^{-1})} \leqslant 1+N(Y)$$

5. (a) $$\frac{N(A^{-1}-X)}{N(A^{-1})} \leqslant \frac{N(XY)}{N(X)} \cdot \frac{1+N(Y)}{1-N(Y)}$$

   (b) $$\frac{N(A^{-1}-X)}{N(A^{-1})} \leqslant N(Y) \cdot \frac{1+N(Y)}{1-N(Y)}$$

6. $$\frac{N(E)}{2N(A)} \cdot \frac{1-N(Y)}{N(X)} \leqslant \frac{N(A^{-1}-X)}{N(A^{-1})},\ E=(I-XA)-(I-AX)$$

7. $$\frac{N(A^{-1}-X)}{N(A^{-1})} \leqslant N(Y).$$

The derivation of the above is based primarily on the relation $Y=I-AX$ and, after transposing and taking inverses, on the relation $(I-Y)^{-1}=I+Y+Y^2+\ldots$, if $N(Y)<1$, which also insures that the eigenvalues of Y are of modulus less than one. It will be very interesting to note in the following tables the relation between numbers 2(a), an absolute error bound and 7, a relative error bound. Before discussing the results, however, it would be good to describe the programs and test matrices.

The Computer Programs

1. LEQ

A FORTRAN subroutine used to solve the matrix equation $AX=B$ and to evaluate the determinant of A. It was written by Max Goldstein of the AEC Computing and Applied Mathematics Center at the Courant Institute of Mathematical Sciences, New York University. The Gauss elimination method is used. The matrices are normalized row-wise by dividing by the largest element of $A(I, J)$ in that row, then the A matrix is reduced to triangular form by $(N-1)$ transformations using pivotal condensation process after which $X\ (I,\ J)$ is computed by a back-substitution process. This transforms B into X and leaves the product of the diagonal elements as the determinant of A.

2. MIDAS

A FORTRAN and ALGOL package to solve general nonsingular systems of linear algebraic equations, invert matrices, and compute determinants. Error bounds on the solution or inverse are available as an option. It was written by Peter A. Businger of Bell Telephone Laboratories, Inc., Murray Hill, New Jersey. The error bound is a bound on the distance between any element of the true inverse and the corresponding element of the computed inverse (unless the bound equals -1, in which case no bound is available). Gaussian elimination with partial pivoting is used to decompose the $N X N$ input matrix into the product of a lower and an upper triangular matrix (LU decomposition). The magnitude of intermediate results is estimated; in case of alarming growth the program switches to complete pivoting. When solving a system of equations $Ax=b$, the accuracy of the solution obtained from the triangular system is improved by iteration; in the case of matrix inversion the iteration is omitted for the sake of computational efficiency. (We note that the error bound is essentially $N(X)N(Y)/[1-N(Y)]$, $Y=I-AX$.)

3. MINV

A FORTRAN subroutine, one of the IBM System/360 Scientific Subroutine Package. It inverts a general matrix by the standard Gauss-Jordan method. The determinant is also calculated. A determinant of zero indicates a singular matrix.

4. SPINV

A single precision FORTRAN IV program for inverting a matrix or solving a set of linear equations. To a program from the SHARE library (7090–F1 3180INV1 Single Precision Matrix Inversion with Selective Pivoting, written by A. R. Sadaka), Sally T. Peavy, National Bureau of Standards, incorporated accuracy checks. This is also the routine used by INVERT of OMNITAB.

5. SOLVE

A FORTRAN program by Cleve Moler given in the book *Computer Solution of Linear Algebraic Systems* by George Forsythe and Cleve Moler. It uses Gauss elimination with partial pivoting and has a subroutine IMPRUV, which can be called to improve the solution of a linear algebraic system. Appropriate messages for various kinds of singularity are available. It is presently undergoing some changes to increase efficiency in most FORTRAN systems although these changes should not materially alter the numerical behavior [2].

6. LINEQ1

A FORTRAN subroutine used to solve the real matrix equation $AX=B$ and written by David S. Dodson, Department of Computer Sciences, Purdue University. The matrix A is factored into lower and upper triangular matrices L and U and then the equations $LZ=B$ and $UX=Z$ are solved in turn. Double precision accumulation of inner products and iterative refinement are used to improve accuracy.

The Test Matrices

An indication of the degree of difficulty that may be encountered in computing the inverse of a matrix is given by the "condition number" of a matrix. There are many ways of arriving at such a number and the one we shall use is called the P-condition number:

$$P(A)=\left|\frac{\lambda}{\mu}\right|$$

526-390 O - 74 - 2

where λ is an eigenvalue of largest modulus and μ, of smallest. We will give this value for each of our test matrices. Since the execution time for inverting matrices of small size (20 by 20) is rather minimal, roughly two seconds, we used matrices of order 100. This means, of course, that the Hilbert matrix had to be excluded as a test matrix ($\log_e P(H_n) \approx 3.5n$, $n=100$) and hence we used only the following five matrices.

1. A_{100}

This is a 100×100 matrix where $A_k = (1/k)I + J$ and J is the 100×100 matrix of all ones. $P(A_k) = 1 + 100k$. The integer form for use as a test matrix is obviously achieved upon multiplication by k.

$$(kA_k)^{-1} = I - \frac{k}{1+100k} \quad J$$

$$P(A_{100}) = 10001$$

$$\approx .10E+05$$

2. A_{1000}

The same as above except that we change the value of k.

$$P(A_{1000}) = 100001$$

$$\approx .10+06$$

3. A_{10000}

The same as above except that we change the value of k.

$$P(A_{10000}) = 1000001$$

$$\approx .10E+07$$

4. T_{100}

This is a 100×100 tridiagonal matrix with -2 on the diagonal, 1 above and below the diagonal, and 0 elsewhere.

$$T = (t_{ij}),\ t_{ij} = \begin{cases} -2, & i=j \\ 1, & |i-j|=1 \\ 0, & |i-j| \geqslant 2. \end{cases}$$

$$P(T_n) \approx (4/\pi^2)n^2$$

$$\approx .41E+04$$

5. T^2_{100}

This is just the square of the above matrix.

$$P(T^2_{100}) \approx (4/\pi^2)^2\,(n^2)^2$$

$$\approx .16E+08$$

The Results

In tables 1 to 10 we give the information with regard to accuracy, using the two different norms, and in tables 11 to 22, the information concerning time; the last set of tables, 23 to 32, will combine the two pieces of information. We will then discuss the results.

TABLE 1. *Accuracy of results*

Program		Max elt of check vector*	$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	$\frac{N(X)N(Y)}{1-N(Y)}$	Relative error	$N(I-XA)$	Rank
LEQ		.68 E−05	.18 E−02	.45 E−05	.18 E−01	.45 E−06	.30 E−02	6
MIDAS		.19 E−03	.24 E−01	.39 E−05	.25 E+00	.41 E−06	.12 E−02	4
MINV		.60 E−06	.79 E−03	.12 E−06	.79 E−02	.12 E−07	.46 E−03	3
SPINV		.79 E−05	.62 E−02	.70 E−05	.61 E−01	.71 E−06	.61 E−02	7
SOLVE	No IMPRUV	.13 E−05	.16 E−02	.42 E−05	.16 E−01	.43 E−06	.33 E−02	5
	IMPRUV	.86 E−07	.22 E−04	.69 E−07	.22 E−03	.70 E−08	.26 E−04	2
LINEQ1		.15 E−06	.25 E−04	.46 E−07	.25 E−03	.47 E−08	.32 E−04	1

Matrix $=A_{100}$
Condition Number $=.10$ E+05
Norm = FROBENIUS

$X=$ Approximate inverse $\quad N(A^{-1}-X) \leq \frac{N(XY)}{1-N(Y)} \leq \frac{N(X)N(Y)}{1-N(Y)}$

*Difference from 1.

$Y=I-AX \quad \frac{N(A^{-1}-X)}{N(A^{-1})} \leq \frac{N(XY)}{1-N(Y)} \cdot \frac{1+N(Y)}{N(X)}$

TABLE 2. *Accuracy of results*

Program		Max elt of check vector*	$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	$\frac{N(X)N(Y)}{1-N(Y)}$	Relative error	$N(I-XA)$	Rank
LEQ		.11 E−03	.20 E−01	.55 E−04	.20 E+00	.56 E−05	.20 E−01	7
MIDAS		.15 E−02	.20 E+00	.11 E−04	.25 E+01	.13 E−05	.19 E−01	5
MINV		.41 E−05	.40 E−01	.41 E−06	.42 E+00	.43 E−07	.39 E−01	3
SPINV		.44 E−03	.62 E−01	.11 E−04	.65 E+00	.11 E−05	.62 E−01	5
SOLVE	No IMPRUV	.69 E−05	.16 E−01	.83 E−05	.16 E+00	.85 E−06	.24 E−01	4
	IMPRUV	.88 E−07	.41 E−03	.13 E−07	.41 E−02	.13 E−08	.47 E−03	1
LINEQ1		.15 E−06	.33 E−03	.32 E−07	.32 E−02	.32 E−08	.33 E−03	2

Matrix $=A_{1000}$
Condition Number $=.10$ E+06
Norm = FROBENIUS

$X=$ Approximate inverse $\quad N(A^{-1}-X) \leq \frac{N(XY)}{1-N(Y)} \leq \frac{N(X)N(Y)}{1-N(Y)}$

*Difference from 1.

$Y=I-AX \quad \frac{N(A^{-1}-X)}{N(A^{-1})} \leq \frac{N(XY)}{1-N(Y)} \cdot \frac{1+N(Y)}{N(X)}$

TABLE 3. *Accuracy of results*

Program		Max elt of check vector*	$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	$\frac{N(X)N(Y)}{1-N(Y)}$	Relative error	$N(I-XA)$	Rank
LEQ		.23 E−02	.19 E+00	.13 E−02	.23 E+01	.16 E−03	.21 E+00	6
MIDAS		.12 E−01	.46 E+01				.18 E+00	
MINV		.45 E−05	.45 E+00	.46 E−06	.81 E+01	.68 E−07	.44 E+00	3
SPINV		.34 E−05	.61 E+00	.26 E−03	.16 E+02	.42 E−04	.61 E+00	5
SOLVE	No IMPRUV	.38 E−05	.16 E+00	.12 E−03	.19 E+01	.14 E−04	.24 E+00	4
	IMPRUV	.00	.33 E−02	.76 E−08	.33 E−01	.77 E−09	.54 E−02	1
LINEQL		.20 E−06	.11 E−01	.19 E−07	.11 E+00	.19 E−08	.11 E−01	2

Matrix $= A_{10000}$
Condition Number $= .10$ E+07
Norm = FROBENIUS

*Difference from 1.

$X =$ Approximate inverse

$Y = I - AX$

$$N(A^{-1}-X) \leqslant \frac{N(XY)}{1-N(Y)} \leqslant \frac{N(X)N(Y)}{1-N(Y)}$$

$$\frac{N(A^{-1}-X)}{N(A^{-1})} \leqslant \frac{N(XY)}{1-N(Y)} \cdot \frac{1+N(Y)}{N(X)}$$

TABLE 4. *Accuracy of results*

Program		Max elt of check vector*	$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	$\frac{N(X)N(Y)}{1-N(Y)}$	Relative error	$N(I-XA)$	Rank
LEQ		.14 E−03	.19 E−04	.16 E−01	.21 E−01	.15 E−04	.57 E−04	4
MIDAS		.14 E−03	.29 E−04	.16 E−01	.31 E−01	.15 E−04	.28 E−04	4
MINV		.75 E−05	.18 E−04	.69 E−03	.19 E−01	.64 E−06	.16 E−04	3
SPINV		.14 E−03	.55 E−04	.16 E−01	.59 E−01	.15 E−04	.52 E−04	4
SOLVE	No IMPRUV	.10 E−04	.19 E−04	.16 E−01	.21 E−01	.15 E−04	.57 E−04	4
	IMPRUV	.15 E−06	.86 E−05	.69 E−05	.93 E−02	.65 E−08	.86 E−05	1
LINEQ1		.15 E−06	.86 E−05	.69 E−05	.93 E−02	.65 E−08	.86 E−05	1

Matrix $= T_{100}$
Condition Number $= .41$ E+04
Norm = FROBENIUS

*Difference from 1.

$X =$ Approximate inverse

$Y = I - AX$

$$N(A^{-1}-X) \leqslant \frac{N(XY)}{1-N(Y)} \leqslant \frac{N(X)N(Y)}{1-N(Y)}$$

$$\frac{N(A^{-1}-X)}{N(A^{-1})} \leqslant \frac{N(XY)}{1-N(Y)} \cdot \frac{1+N(Y)}{N(X)}$$

TABLE 5. *Accuracy of results*

Program		Max elt of check vector*	$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	$\frac{N(X)N(Y)}{1-N(Y)}$	Relative error	$N(I-XA)$	Rank
LEQ		.60 E+00	.83 E−01	.82 E+05	.10 E+06	.78 E−01	.20 E+03	6
MIDAS		.29 E+00	.10 E+00	.39 E+05	.12 E+06	.39 E−01	.74 E+02	5
MINV		.21 E+00	.15 E+00	.27 E+05	.18 E+06	.30 E−01	.16 E+00	3
SPINV		.26 E+00	.27 E+03				.24 E+00	
SOLVE	No IMPRUV	.14 E−01	.13 E+00	.36 E+05	.17 E+06	.37 E−01	.18 E+03	4
	IMPRUV	.00	.30 E−01	.70 E−02	.34 E+05	.67 E−08	.30 E−01	1
LINEQ1		.31 E−05	.29 E−01	.43 E+00	.32 E+05	.41 E−06	.30 E−01	2

Matrix $= T^2_{100}$
Condition Number = .16 E+08
Norm = FROBENIUS

*Difference from 1.

X = Approximate inverse

$Y = I - AX$

$$N(A^{-1}-X) \leq \frac{N(XY)}{1-N(Y)} \leq \frac{N(X)N(Y)}{1-N(Y)}$$

$$\frac{N(A^{-1}-X)}{N(A^{-1})} \leq \frac{N(XY)}{1-N(Y)} \cdot \frac{1+N(Y)}{N(X)}$$

TABLE 6. *Accuracy of results*

Program		Max elt of check vector*	$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	$\frac{N(X)N(Y)}{1-N(Y)}$	Relative error	$N(I-XA)$	Rank
LEQ		.68 E−05	.31 E−02	.67 E−04	.30 E+00	.68 E−06	.21 E−01	4
MIDAS		.19 E−03	.24 E+00	.17 E−03	.31 E+02	.21 E−05	.33 E−02	7
MINV		.60 E−06	.69 E−02	.81 E−06	.69 E+00	.82 E−08	.55 E−03	3
SPINV		.79 E−05	.92 E−02	.13 E−03	.92 E+00	.13 E−05	.89 E−02	6
SOLVE	No IMPRUV	.13 E−05	.25 E−02	.84 E−04	.24 E+00	.85 E−06	.26 E−01	5
	IMPRUV	.86 E−07	.46 E−04	.69 E−06	.45 E−02	.70 E−08	.49 E−04	1
LINEQ1		.15 E−06	.69 E−04	.69 E−06	.68 E−02	.70 E−08	.79 E−04	1

Matrix $= A_{100}$
Condition Number = .10 E+05
Norm $= n \cdot$ (Maximum Element)

*Difference from 1.

X = Approximate inverse

$Y = I - AX$

$$N(A^{-1}-X) \leq \frac{N(XY)}{1-N(Y)} \leq \frac{N(X)N(Y)}{1-N(Y)}$$

$$\frac{N(A^{-1}-X)}{N(A^{-1})} \leq \frac{N(XY)}{1-N(Y)} \cdot \frac{1+N(Y)}{N(X)}$$

TABLE 7. *Accuracy of results*

Program		Max elt of check vector*	$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	$\frac{N(X)N(Y)}{1-N(Y)}$	Relative error	$N(I-XA)$	Rank
LEQ		.11 E−03	.37 E−01	.58 E−03	.38 E+01	.61 E−05	.79 E−01	4
MIDAS		.15 E−02	.20 E+01				.31 E−01	
MINV		.40 E−05	.16 E+00	.18 E−05	.19 E+02	.21 E−07	.39 E−01	3
SPINV		.44 E−03	.92 E−01	.81 E−03	.10 E+02	.90 E−05	.92 E−01	6
SOLVE	No IMPRUV	.69 E−05	.29 E−01	.74 E−03	.29 E+01	.77 E−05	.12 E+00	5
	IMPRUV	.88 E−07	.62 E−03	.11 E−06	.62 E−01	.11 E−08	.70 E−03	1
LINEQ1		.15 E−06	.97 E−03	.64 E−06	.96 E−01	.65 E−08	.97 E−03	2

Matrix $=A_{1000}$
Condition Number $=.10$ E+06
Norm $=n\cdot$ (Maximum Element)

*Difference from 1.

$X=$ Approximate inverse

$Y=I-AX$

$$N(A^{-1}-X)\leq\frac{N(XY)}{1-N(Y)}\leq\frac{N(X)N(Y)}{1-N(Y)}$$

$$\frac{N(A^{-1}-X)}{N(A^{-1})}\leq\frac{N(XY)}{1-N(Y)}\cdot\frac{1+N(Y)}{N(X)}$$

TABLE 8. *Accuracy of results*

Program		Max elt of check vector*	$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	$\frac{N(X)N(Y)}{1-N(Y)}$	Relative error	$N(I-XA)$	Rank
LEQ		.23 E−02	.30 E+00	.15 E−01	.43 E+02	.20 E−03	.76 E+00	4
MIDAS		.12 E−01	.46 E+02				.31 E+00	
MINV		.45 E−05	.15 E+01				.44 E+00	
SPINV		.34 E−05	.86 E+00	.71 E−01	.62 E+03	.13 E−02	.88 E+00	5
SOLVE	No IMPRUV	.38 E−05	.22 E+00	.13 E−01	.27 E+02	.15 E−03	.16 E+01	3
	IMPRUV	.00	.50 E−02	.47 E−07	.50 E+00	.47 E−09	.11 E−01	1
LINEQ1		.20 E−06	.12 E−01	.80 E−06	.12 E+01	.82 E−08	.12 E−01	2

Matrix $=A_{10000}$
Condition Number $=.10$ E+07
Norm $=n\cdot$ (Maximum Element)

*Difference from 1.

$X=$ Approximate inverse

$Y=I-AX$

$$N(A^{-1}-X)\leq\frac{N(XY)}{1-N(Y)}\leq\frac{N(X)N(Y)}{1-N(Y)}$$

$$\frac{N(A^{-1}-X)}{N(A^{-1})}\leq\frac{N(XY)}{1-N(Y)}\cdot\frac{1+N(Y)}{N(X)}$$

TABLE 9. *Accuracy of results*

Program		Max elt of check vector*	$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	$\frac{N(X)N(Y)}{1-N(Y)}$	Relative error	$N(I-XA)$	Rank
LEQ		.14 E−03	.12 E−03	.33 E−01	.30 E+00	.13 E−04	.36 E−03	4
MIDAS		.14 E−03	.14 E−03	.32 E−01	.36 E+00	.13 E−04	.13 E−03	4
MINV		.75 E−05	.12 E−03	.14 E−02	.30 E+00	.55 E−06	.72 E−04	3
SPINV		.14 E−03	.33 E−03	.33 E−01	.84 E+00	.13 E−04	.36 E−03	4
SOLVE	No IMPRUV	.10 E−04	.12 E−03	.33 E−01	.30 E+00	.13 E−04	.36 E−03	4
	IMPRUV	.15 E−06	.36 E−04	.24 E−04	.90 E−01	.94 E−08	.36 E−04	1
LINEQ1		.15 E−06	.36 E−04	.24 E−04	.90 E−01	.94 E−08	.36 E−04	1

Matrix $= T_{100}$
Condition Number $= .41$ E+04
Norm $= n \cdot$ (Maximum Element)

*Difference from 1.

$X =$ Approximate inverse

$Y = I - AX$

$$N(A^{-1}-X) \leq \frac{N(XY)}{1-N(Y)} \leq \frac{N(X)N(Y)}{1-N(Y)}$$

$$\frac{N(A^{-1}-X)}{N(A^{-1})} \leq \frac{N(XY)}{1-N(Y)} \cdot \frac{1+N(Y)}{N(X)}$$

TABLE 10. *Accuracy of results*

Program		Max elt of check vector*	$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	$\frac{N(X)N(Y)}{1-N(Y)}$	Relative error	$N(I-XA)$	Rank
LEQ		.60 E+00	.88 E+00	.12 E+07	.17 E+08	.10 E+01	.11 E+04	5
MIDAS		.29 E+00	.71 E+00	.24 E+06	.54 E+07	.18 E+00	.48 E+03	4
MINV		.21 E+00	.76 E+00	.19 E+06	.65 E+07	.16 E+00	.93 E+00	3
SPINV		.26 E+00	.14 E+04				.12 E+01	
SOLVE	No IMPRUV	.14 E−01	.22 E+01				.13 E+04	
	IMPRUV	.00	.17 E+00	.29 E−01	.44 E+06	.16 E−07	.17 E+00	1
LINEQ1		.31 E−05	.15 E+00	.93 E+00	.37 E+06	.53 E−06	.17 E+00	2

Matrix $= T^2_{100}$
Condition Number $= .16$ E+08
Norm $= n$. (Maximum Element)

*Difference from 1.

$X =$ Approximate inverse

$Y = I - AX$

$$N(A^{-1}-X) \leq \frac{N(XY)}{1-N(Y)} \leq \frac{N(X)N(Y)}{1-N(Y)}$$

$$\frac{N(A^{-1}-X)}{N(A^{-1})} \leq \frac{N(XY)}{1-N(Y)} \cdot \frac{1+N(Y)}{N(X)}$$

NOTE: In tables 11 to 17, the increase in time for calculating the error bounds for T^2_{100} is due to the fact that the exact inverse was also calculated and used in evaluating the results for that matrix. Although these figures are not listed in this paper, they once again show the value of using $N(XY)$ instead of $N(X)N(Y)$.

TABLE 11. *Breakdown of run time in seconds per program*

Matrix	Set up	Solution	Error bounds	Total
A_{100}	6.4	17.6	59.4	83.4
A_{1000}	6.4	17.4	58.7	82.6
A_{10000}	6.4	17.6	58.6	82.6
T_{100}	3.8	17.4	58.6	79.9
T^2_{100}	18.1	17.6	85.7	121.4

Program = LEQ

TABLE 12. *Breakdown of run time in seconds per program*

Matrix	Set up	Solution	Error bounds	Total
A_{100}	6.3	49.1	59.6	108.0
A_{1000}	6.3	41.7	59.0	107.0
A_{10000}	6.2	42.6	60.2	109.0
T_{100}	3.8	42.3	59.7	105.8
T^2_{100}	18.3	42.6	86.5	147.5

Program = MIDAS

TABLE 13. *Breakdown of run time in seconds per program*

Matrix	Set up	Solution	Error bounds	Total
A_{100}	8.4	29.3	61.1	98.7
A_{1000}	6.4	29.4	59.7	95.5
A_{10000}	6.5	29.4	59.6	95.5
T_{100}	3.8	29.9	58.8	92.5
T^2_{100}	18.3	30.0	86.6	134.8

Program = MINV

TABLE 14. *Breakdown of run time in seconds per program*

Matrix	Set up	Solution	Error bounds	Total
A_{100}	6.3	42.1	59.3	107.7
A_{1000}	6.4	42.0	59.2	107.6
A_{10000}	6.3	42.1	59.4	107.7
T_{100}	3.7	41.9	58.5	104.0
T^2_{100}	18.2	42.7	85.5	146.3

Program = SPINV

TABLE 15. *Breakdown of run time in seconds per program*

Matrix	Set up	Solution	Error bounds	Total
A_{100}	6.3	10.7	59.8	76.8
A_{1000}	6.3	10.6	59.7	76.7
A_{10000}	6.2	10.4	58.9	75.6
T_{100}	3.7	10.4	52.9	67.1
T^2_{100}	18.2	10.6	87.1	115.9

Program = SOLVE (No IMPRUV)

TABLE 16. *Breakdown of run time in seconds per program*

Matrix	Set up	Solution	Error bounds	Total
A_{100}	6.4	55.7	60.1	122.2
A_{1000}	6.3	55.6	59.1	121.0
A_{10000}	6.2	54.6	58.9	119.8
T_{100}	3.8	55.6	58.5	117.8
T^2_{100}	14.9	136.7	84.8	236.4

Program = SOLVE (With IMPRUV)

TABLE 17. *Breakdown of run time in seconds per program*

Matrix	Set up	Solution	Error bounds	Total
A_{100}	6.2	49.0	59.6	114.9
A_{1000}	6.4	49.2	59.6	115.2
A_{10000}	6.4	49.8	60.1	116.3
T_{100}	3.8	49.6	57.1	110.5
T^2_{100}	18.3	79.8	84.9	183.0

Program = LINEQ1

TABLE 18. *Comparison of programs according to time in seconds*

Program		Set up	Solution	Error bounds	Total	Rank
LEQ		6.4	17.6	59.4	83.4	2
MIDAS		6.3	42.1	59.6	108.0	4
MINV		8.4	29.3	61.1	98.7	3
SPINV		6.3	42.1	59.3	107.7	4
SOLVE	No IMPRUV	6.3	10.7	59.8	76.8	1
	IMPRUV	6.4	55.7	60.1	122.2	7
LINEQ1		6.2	49.0	59.6	114.9	6

Matrix $= A_{100}$ Condition Number $= .10\mathrm{E}+04$

TABLE 19. *Comparison of programs according to time in seconds*

Program		Set up	Solution	Error bounds	Total	Rank
LEQ		6.4	17.4	58.7	82.6	2
MIDAS		6.3	41.7	59.0	107.0	4
MINV		6.4	29.4	59.7	95.5	3
SPINV		6.4	42.0	59.2	107.6	5
SOLVE	No IMPRUV	6.3	10.6	59.7	76.7	1
	IMPRUV	6.3	55.6	59.1	121.0	7
LINEQ1		6.4	49.2	59.6	115.2	6

Matrix $= A_{1000}$ Condition Number $= .10\ \mathrm{E}+05$

TABLE 20. *Comparison of programs according to time in seconds*

Program		Set up	Solution	Error bounds	Total	Rank
LEQ		6.4	17.6	58.6	82.6	2
MIDAS		6.2	42.6	60.2	109.0	5
MINV		6.5	29.4	59.6	95.5	3
SPINV		6.3	42.1	59.4	107.7	4
SOLVE	No IMPRUV	6.2	10.4	58.9	75.6	1
	IMPRUV	6.2	54.6	58.9	119.8	7
LINEQ1		6.4	49.8	60.1	116.3	6

Matrix $= A_{10000}$ Condition Number $= .10\ \mathrm{E}+06$

TABLE 21. *Comparison of programs according to time in seconds*

Program		Set up	Solution	Error bounds	Total	Rank
LEQ		3.8	17.4	58.6	79.9	2
MIDAS		3.8	42.3	59.7	105.8	5
MINV		3.8	29.9	58.8	92.5	3
SPINV		3.7	41.9	58.5	104.0	4
SOLVE	No IMPRUV	3.7	10.4	52.9	67.1	1
	IMPRUV	3.8	55.6	58.5	117.8	7
LINEQ1		3.8	49.6	57.1	110.5	6

Matrix $= T_{100}$ Condition Number $= .41$ E+05

TABLE 22. *Comparison of programs according to time in seconds*

Program		Set up	Solution	Error bounds	Total	Rank
LEQ		18.1	17.6	85.7	121.4	2
MIDAS		18.3	42.6	86.5	147.5	4
MINV		18.3	30.0	86.6	134.8	3
SPINV		18.2	42.7	N(Y) > 1		
SOLVE	No IMPRUV	18.2	10.6	87.1	115.9	1
	IMPRUV	14.9	136.7	84.8	236.4	6
LINEQ1		18.3	79.8	84.9	183.0	5

Matrix $= T^2_{100}$ Condition Number $= .16$ E+08

TABLE 23. *Summary of results*

Program		$N(I-AX)$	$\dfrac{N(XY)}{1-N(Y)}$	Subroutine time (Seconds)
LEQ		.18 E−02	.45 E−05	17.6
MIDAS		.24 E−01	.39 E−05	42.1
MINV		.79 E−03	.12 E−06	29.3
SPINV		.62 E−02	.70 E−05	42.1
SOLVE	No IMPRUV	.16 E−02	.42 E−05	10.7
	IMPRUV	.22 E−04	.69 E−07	55.7
LINEQ1		.25 E−04	.46 E−07	49.0

Matrix $= A_{100}$
Condition Number $= .10$ E+04
Norm = FROBENIUS

$$N(A^{-1}-X) \leq \frac{N(XY)}{1-N(Y)}$$

TABLE 24. *Summary of results*

Program		$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	Subroutine time (Seconds)
LEQ		.20 E−01	.55 E−04	17.4
MIDAS		.20 E+00	.11 E−04	41.7
MINV		.40 E−01	.41 E−06	29.4
SPINV		.62 E−01	.11 E−04	42.0
SOLVE	No IMPRUV	.16 E−01	.83 E−05	10.6
	IMPRUV	.41 E−03	.13 E−07	55.6
LINEQ1		.33 E−03	.32 E−07	49.2

Matrix $= A_{1000}$
Condition Number $=$ .10 E+05
Norm $=$ FROBENIUS

$$N(A^{-1}-X) \leq \frac{N(XY)}{1-N(Y)}$$

TABLE 25. *Summary of results*

Program		$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	Subroutine time (Seconds)
LEQ		.19 E+00	.13 E−02	17.6
MIDAS		.46 E+01		
MINV		.45 E+00	.46 E−06	29.4
SPINV		.61 E+00	.26 E−03	42.1
SOLVE	No IMPRUV	.16 E+00	.12 E−03	10.4
	IMPRUV	.33 E−02	.76 E−08	54.6
LINEQ1		.11 E−01	.19 E−07	49.8

Matrix $= A_{10000}$
Condition Number $=$ .10 E+06
Norm $=$ FROBENIUS

$$N(A^{-1}-X) \leq \frac{N(XY)}{1-N(Y)}$$

TABLE 26. *Summary of results*

Program		$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	Subroutine time (Seconds)
LEQ		.19 E−04	.16 E−01	17.4
MIDAS		.29 E−04	.16 E−01	42.3
MINV		.18 E−04	.69 E−03	29.9
SPINV		.55 E−04	.16 E−01	41.9
SOLVE	No IMPRUV	.19 E−04	.16 E−01	10.4
	IMPRUV	.86 E−05	.69 E−05	55.6
LINEQ1		.86 E−05	.69 E−05	49.6

Matrix $= T_{100}$
Condition Number $=$ .41 E+05
Norm $=$ FROBENIUS

$$N(A^{-1}-X) \leq \frac{N(XY)}{1-N(Y)}$$

TABLE 27. *Summary of results*

Program		$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	Subroutine time (Seconds)
LEQ		.83 E − 01	.82 E + 05	17.6
MIDAS		.10 E + 00	.39 E + 05	42.6
MINV		.15 E + 00	.27 E + 05	30.0
SPINV		.27 E + 03		
SOLVE	No IMPRUV	.13 E + 00	.36 E + 05	10.6
	IMPRUV	.30 E − 01	.70 E − 02	136.7
LINEQ1		.29 E − 01	.43 E + 00	79.8

Matrix $= T_{100}^2$
Condition Number $= .16$ E + 08
Norm = FROBENIUS

$$N(A^{-1}-X) \leqslant \frac{N(XY)}{1-N(Y)}$$

TABLE 28. *Summary of results*

Program		$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	Subroutine time (Seconds)
LEQ		.31 E − 02	.67 E − 04	17.6
MIDAS		.24 E + 00	.17 E − 03	42.1
MINV		.69 E − 02	.81 E − 06	29.3
SPINV		.92 E − 02	.13 E − 03	42.1
SOLVE	No IMPRUV	.25 E − 02	.84 E − 04	10.7
	IMPRUV	.46 E − 04	.69 E − 06	55.7
LINEQ1		.69 E − 04	.69 E − 06	49.0

Matrix $= A_{100}$
Condition Number $= .10$ E + 04
Norm $= n \cdot$ (Maximum Element)

$$N(A^{-1}-X) \leqslant \frac{N(XY)}{1-N(Y)}$$

TABLE 29. *Summary of results*

Program		$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	Subroutine time (Seconds)
LEQ		.37 E−01	.58 E−03	17.4
MIDAS		.20 E+01		
MINV		.16 E+00	.18 E−05	29.4
SPINV		.92 E−01	.81 E−03	42.0
SOLVE	No IMPRUV	.29 E−01	.74 E−03	10.6
	IMPRUV	.62 E−03	.11 E−06	55.6
LINEQ1		.97 E−03	.64 E−06	49.2

Matrix $= A_{1000}$
Condition Number $= .10$ E $+05$
Norm $= n \cdot$ (Maximum Element)

$$N(A^{-1}-X) \leqslant \frac{N(XY)}{1-N(Y)}$$

TABLE 30. *Summary of results*

Program		$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	Subroutine time (Seconds)
LEQ		.30 E+00	.15 E−01	17.6
MIDAS		.46 E+02		
MINV		.15 E+01		
SPINV		.86 E+00	.71 E−01	42.1
SOLVE	No IMPRUV	.22 E+00	.13 E−01	10.4
	IMPRUV	.50 E−02	.47 E−07	54.6
LINEQ1		.12 E−01	.80 E−06	49.8

Matrix $= A_{10000}$
Condition Number $= .10$ E $+06$
Norm $= n \cdot$ (Maximum Element)

$$N(A^{-1}-X) \leqslant \frac{N(XY)}{1-N(Y)}$$

TABLE 31. *Summary of results*

Program		$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	Subroutine time (Seconds)
LEQ		.12 E−03	.33 E−01	17.4
MIDAS		.14 E−03	.32 E−01	42.3
MINV		.12 E−03	.14 E−02	29.9
SPINV		.33 E−03	.33 E−01	41.9
SOLVE	No IMPRUV	.12 E−03	.33 E−01	10.4
	IMPRUV	.36 E−04	.24 E−04	55.6
LINEQ1		.36 E−04	.24 E−04	49.6

Matrix $=T_{100}$
Condition Number = .41 E+05
Norm $=n\cdot$ (Maximum Element)

$$N(A^{-1}-X)\leq\frac{N(XY)}{1-N(Y)}$$

TABLE 32. *Summary of results*

Program		$N(I-AX)$	$\frac{N(XY)}{1-N(Y)}$	Subroutine time (Seconds)
LEQ		.88 E+00	.12 E+07	17.6
MIDAS		.71 E+00	.24 E+06	42.6
MINV		.76 E+00	.19 E+06	30.0
SPINV		.14 E+04		
SOLVE	No IMPRUV	.22 E+01		
	IMPRUV	.17 E+00	.29 E−01	136.7
LINEQ1		.15 E+00	.93 E+00	79.8

Matrix $=T^2_{100}$
Condition Number = .16 E+08
Norm $=n\cdot$ (Maximum Element)

$$N(A^{-1}-X)\leq\frac{N(XY)}{1-N(Y)}$$

Discussion

As was mentioned earlier, the value of multiplying matrices before taking the norm of a product of two matrices is clearly demonstrated in tables 1 through 10. $N(I-AX)$ is a relative error bound and $N(XY)/[1-N(Y)]$ is an absolute error bound and yet for the A_k matrices, the latter was always smaller than the former. For the T matrices this is not true except when iterative refinement (IMPRUV) is used. In these cases, however, the relative error in column 5 is a much better bound than $N(I-AX)$. As was indicated in the previous paper also, this absolute error bound is as close to the actual error as one could expect.

Let us now turn our attention to the time element. For small matrices, the use of iterative refinement added such a small increase (1 second for the 20×20 case) that it seems definitely useful. For larger matrices, however, the picture is not quite so clear, but let us make some general observations first.

As will be noticed in tables 11 to 22, the information that took longest to gather was the error bounds. It is not necessary to calculate all this information in a particular run but only what would be useful. What is included in that part of the program is the computation of $I-AX$, $I-XA$, $X(I-AX)$, the difference between the residuals and the calculation of the two norms for these quantities. It is up to the user to decide what is the necessary information.

From tables 1 to 10 it can be seen that the programs without iterative refinement performed quite similarly concerning accuracy, with MINV consistently being slightly better. Iterative refinement, of course, had its desirable effect. From tables 11 to 17 we see that each program was consistent in its execution time for the different matrices with the exception of the two programs that used iterative refinement, LINEQ1 and SOLVE with IMPRUV. In these programs more iterations were needed for the most ill-conditioned matrix, T^2_{100}.

In tables 18 to 22 it can be seen that SOLVE without IMPRUV was definitely the fastest. As a point of information for those familiar with this program, we used the new version given in (2) to find the inverse of T_{100}. The times are given in the following table:

	Old version	New version
DECOMP................	2.7	.3
SOLVE..................	8.2	8.7
Total......................	10.9	9.0

Admittedly, the times are rather minimal but the decrease for DECOMP is considerable. Whether this reduction is primarily due to omitting the scaling used in the old version or to the different way of writing matrix multiplication is not clear. However, abundant support for the latter is given in (2) and this increase in efficiency makes SOLVE without IMPRUV even faster than the other programs. We might add that the numerical accuracy did not change: all digits were identical in both runs.

The two programs using iterative refinement were quite comparable except in the case of T^2_{100}. For this matrix, LINEQ1 had 6 digit accuracy whereas SOLVE with IMPRUV had 8 digit accuracy in almost every element.

The more important question of whether iterative refinement is worth the extra time remains. This is an almost impossible question to answer in the abstract. The proposer of the problem is really the only one who can make that decision. If an accurate inverse in itself is the desired end-product, then some criterion for $N(A^{-1}-X)$ may be used to decide. (It is certainly important that this criterion be included in the output of *every* program anyway.) It seems from tables 23 to 35 that for A_{100}, A_{1000}, and A_{10000} iterative refinement would not be needed and that SOLVE without IMPRUV would be the most efficient—an excellent error bound in the fastest time. It would seem that for T^2_{100} some improvement *is* necessary. However, to let the program run its full length might not be necessary. From our experience with SOLVE with IMPRUV using a UNIVAC 1108, we estimate for this size matrix approximately 20–25 seconds per iteration and each iteration yields at least one digit improvement. The way SOLVE is set up allows the user to decide whether or not to use the subroutine IMPRUV and, if used, the maximum number of iterations to be performed. Approximation of the number of correct digits in the nonimproved computed inverse is also available from one iteration in this subroutine. (See [3], p. 50.) We have found those estimates to be very good.

In summary, for programs without iterative refinement, it seems that SOLVE is the fastest and MINV, the most accurate. It also seems that SOLVE has the best combination of accuracy and time. If iterative refinement may be desired, it seems the optional nature of IMPRUV and its added information would indicate its high value.

It still remains up to the originator of the problem to decide just exactly what is desired. At any rate, the purpose of the information contained herein is to help whomever has to make the decision.

References

[1] Fitzgerald, Kenneth E., Error estimates for the solution of linear algebraic systems, J. Res. Nat. Bur. Stand. (U.S.) **74B** (Math. Sci.), No. 4, 251–310 (Oct.–Dec. 1970).

[2] Moler, Cleve B., Matrix computations with FORTRAN and paging, Comm. ACM **15**, 4 (April 1972), 268–270, 274.

[3] Forsythe, George, and Moler, Cleve B., Computer Solution of Linear Algebraic Systems (Prentice-Hall, Inc., Englewood Cliffs, N.J., 1967).

(Paper 78B1–395)

526-390 O - 74 - 3