Technical Report CS-93-08

Department of Computer Systems
Faculty of Mathematics and Computer Science
University of Amsterdam

Stability of Gauss-Huard Elimination for Solving Linear Systems

T. J. Dekker
W. Hoffmann
K. Potma

August 1993

$$I - v_k e_k^T = \begin{pmatrix} 1 & & x & & & \\ & 1 & x & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \end{pmatrix}, \quad D_k^{-1}(I - e_k h_k^T) = \begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ x & x & x & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \end{pmatrix}$$

Stability of Gauss-Huard Elimination for Solving Linear Systems

T. J. Dekker, W. Hoffmann, K. Potma
Department of Computer Systems
University of Amsterdam

Abstract
This paper considers elimination methods to solve dense linear systems, in particular a variant due to Huard of Gaussian elimination [13]. This variant reduces the system to an equivalent diagonal system just as Gauss-Jordan elimination, but does not require more floating-point operations than Gaussian elimination. Huard's method may be advantageous for use in computers with hierarchical memory, such as cache, and in distributed-memory systems.

An error analysis is given showing that Huard's elimination method is as stable as Gauss-Jordan elimination with appropriate pivoting strategy. This result was announced in [5] and is proven in a similar way as the proof of stability for Gauss-Jordan given in [4].

Keywords
Gaussian elimination, Gauss-Jordan, Linear systems, LU factorization, pivoting strategies, parallel algorithms, vector computing.

1. Introduction

Gaussian elimination still seems to be the most popular method to solve dense linear systems. It requires roughly $2/3\ n^3$ floating-point operations and can be organized such that it performs well on vector processors and parallel systems [9] [12] [16].

Two other elimination algorithms which reduce the given matrix not to triangular but to *diagonal* form (or the identity matrix) are the following. Gauss-Jordan elimination requires about n^3 floating-point operations, which is 50 % more than Gaussian elimination. It is however very suitable for vector processors, because it operates on full n-vectors (not decreasing in length at each elimination step) and it requires the same total number of vector operations as Gaussian elimination. Thus only for very large systems Gaussian elimination is faster on vector computers [9]. Moreover, Gauss-Jordan with an appropriate pivoting strategy is fairly stable [4].
Another variant due to Huard [13], also treated by Cosnard et al. [2], also reduces the given matrix to diagonal form, as Gauss-Jordan does, but surprisingly requires only about $2/3\ n^3$ floating-point operations, the same as for Gaussian elimination. Implementation of this variant on a vector computer yielded fully satisfactory results. The method appears to be numerically stable and to be competitive in speed with Gaussian elimination; in particular, Huard's variant may, for large n, be faster on a shared-memory system, because it requires less cache memory on average. This appears from experiments by Hoffmann [11] and Potma [15]. Further research is continued on the performance of these algorithms on various parallel systems.

In this paper we show that Huard's method is as stable as Gauss-Jordan with the same appropriate pivoting strategy, which result we have announced in [5].

In the next section we briefly describe Gaussian elimination and (known) results of its error analysis. In section 3 we similarly describe Gauss-Jordan and results of error analysis. In section 4 we describe Gauss-Huard and give a proof of our new result on error analysis.

Notations and preliminaries.
We consider given n-th order matrix A and given right-hand side vector b, for which we want to find vector x solving the linear system

(1.1) $$A\ x = b.$$

Elimination algorithms are mostly performed with pivoting, which in its most general form may entail both row and column interchanges. Let P denote the permutation matrix of the row interchanges and Q the permutation matrix of the column interchanges. Accordingly, we define

(1.2) $A' := PAQ, \quad b' := Pb, \quad x' := Q^{-1}x.$

Thus, system (1.1) is equivalent with the system

(1.3) $A'\ x' = b'$

and any algorithm with pivoting can be viewed as applying the corresponding basic algorithm without interchanges to this equivalent system.

In the theorems and proofs on error analysis given below we use the following notations. Absolute values and comparisons of vectors and matrices are to be understood elementwise. O(n) denotes some appropriate small multiple of n, and ε is the machine precison.
We assume that, with this machine precision, the arithmetic satisfies Wilkinson's axioms for floating-point arithmetic [19] [8], and that no arithmetic exceptions, such as overflow and underflow occur in the calculations considered. Moreover, as usual, we assume $n\,\varepsilon < 1$. Thus, ε is small enough to ensure that the higher order terms in the error analysis, $O(n^2\varepsilon^2)$ and higher, can be subsumed in the main terms given. Therefore, we always omit the higher order terms.

2. Gaussian Elimination

Gaussian elimination consists of three parts, namely LU factorization and forward and back substitution.

LU factorization with pivoting can be described as: *find lower triangular L, upper triangular U and suitable permutation matrices P and Q such that*

(2.1) $PAQ = LU.$

Thus the LU factorization can be considered as starting from matrix A' = PAQ, i.e. we may describe the process as if the row and/or column permutations have been performed beforehand, and the pivot in each elimination step is in the proper diagonal position. Matrix A' is then reduced to an upper triangular matrix, U, by means of n-1 successive elimination steps.

In fact the diagonals of L and U can here be chosen in different ways. The most general form of the decomposition is LDU, where D is an appropriate diagonal matrix [9]. For simplicity, we have taken D = I (i.e. the identity matrix) and have chosen L (as usual) *unit* lower triangular, i.e. the main diagonal elements of L are equal to one.

Thus, LU factorization reduces the given system to the equivalent system

(2.2) $LU\ x' = b',$

which is subsequently solved by forward and back substitution.
Forward substitution solves the lower triangular system

(2.3) $L\ y = b'$

and back substitution solves the upper triangular system

(2.4) $Ux' = y.$

Pivot selection strategies
Gaussian elimination is mostly performed with pivoting, i.e. selecting a matrix element of sufficiently large magnitude as pivot of the elimination process, and correspondingly interchanging rows and/or columns of the matrix to bring the pivot in proper position. Partial pivoting can be either with *row* interchanges only (then Q = I) or with *column* interchanges only (then P = I). Complete pivoting requires both row and column interchanges.

The pivoting is practically always needed to ensure numerical stability. In direct methods for sparse matrices, it is also used to reduce fill-in in the factor matrices L and U, see Duff et al. [9]. Mostly, partial pivoting is

performed, and complete pivoting is seldom used. For large systems, however, numerical stability can only be guaranteed with complete pivoting, as shown by Wilkinson [18].
Complete pivoting requires about 50 % more operations (namely comparisons) than partial pivoting. Moreover, it requires a substantial communication overhead on distributed parallel systems. It is possible, however, to combine partial and complete pivoting as follows. Partial pivoting is performed with *monitoring the pivot growth*, i.e. in each elimination step an upper bound of the pivot growth is calculated and compared with a certain threshold. When this threshold is exceeded, complete pivoting is used in the remaining elimination steps. A careful choice of the threshold parameter ensures that the algorithm practically is as economic as Gaussian elimination with partial pivoting and as reliable as Gaussian elimination with complete pivoting [1] [9]. Moreover, the partial pivoting with monitoring requires only little extra overhead on a distributed-memory system, as appears from experiments by Hoffmann & Potma [12].

Error analysis
The results of error analysis of Gaussian elimination, established by Wilkinson, is well known [18] [19]. We briefly mention these results in a form suitable for our subsequent analysis of Gauss-Huard, similar to the formulation of Golub & Van Loan [8].

Gaussian elimination to solve system (2.1) satisfies the following theorem. Note that we use the notations mentioned in section 1.

2.5. Theorem. *Gaussian elimination with (partial or complete) pivoting, calculating in floating-point arithmetic with machine precision ε, assuming that no arithmetic exceptions occur,* exactly *satisfies the following relations,where E_1, E_2, E_3 are appropriate matrices:*

(2.5.1) $A' + E_1 = LU$, $|E_1| \le \varepsilon\, O(n)\, (|A'| + |L|\,|U|)$, *(LU factorization);*

(2.5.2) $(L + E_2)\, y = b'$, $|E_2| \le \varepsilon\, O(n)\, |L|$, *(forward substitution);*

(2.5.3) $(U + E_3)\, x' = y$, $|E_3| \le \varepsilon\, O(n)\, |U|$, *(back substitution).*

Hence, the total result for Gaussian elimination is as follows:

(2.5.4) $(A' + \delta A')\, x' = (L + E_2)\,(U + E_3)\, x' = b'$,

(2.5.5) $|\delta A'| = |\, E_1 + E_2 U + L E_3 + E_2 E_3\, | \le \varepsilon\, O(n)\, (|A'| + |L|\,|U|)$,

or in terms of matrix norms:

(2.5.6) $||\delta A|| = ||\delta A'|| \le \varepsilon\, O(n)\, (||A|| + ||L||\,||U||)$.

Remark. Wilkinson formulated his error analysis in terms of the growth, g, i.e. the largest magnitude of the elements obtained in the Gaussian elimination process. We then have

$$|L|\,|U| \le n\, g,$$

and we obtain, for any matrix norm:

$$||\delta A|| = ||\delta A'|| \le \varepsilon\, O(n^3) g\, ||A||.$$

For (different forms of) proof of this theorem we refer to [8] [18] [19].

3. Gauss-Jordan elimination

The algorithm of Gauss-Jordan transforms matrix A by means of elementary transformations into a diagonal matrix (or into the identity matrix), and performs similar transformations to the right-hand side vector b in order to find the solution vector x. We first consider the basic elimination algorithm and then the algorithm with pivoting and the pivoting strategy.

Basic Gauss - Jordan algorithm

The transformation of A is achieved in n successive elimination steps. Starting from $A^{(0)} = A' = PAQ$, the k-th elimination step, k = 1, ... , n, transforms $A^{(k-1)}$ into $A^{(k)}$ such that the off-diagonal elements in the k-th column, not only below but also above the main diagonal, become zero. Thus, after n steps the diagonal matrix $D = A^{(n)}$ is obtained.
The k-th elimination step can be formulated as follows.
The pivot of the k-th elimination step is $\delta_k = A^{(k-1)}_{k,k}$, as in Gaussian elimination. Let g_k be the column vector given by

(3.1) $$g_k = A^{(k-1)} e_k - \delta_k e_k ,$$

i.e. the vector obtained from the k-th column of $A^{(k-1)}$ by replacing its diagonal element by zero.
Then the k-th elimination step, to introduce the required zeros in the k-th column of the matrix, consists of premultiplying $A^{(k-1)}$ by the matrix

(3.2) $$T_k = I - \delta_k^{-1} g_k e_k^T.$$

In other words, $A^{(k)}$ is obtained from $A^{(k-1)}$ as follows

(3.3) $$A^{(k)} = T_k A^{(k-1)} = A^{(k-1)} - \delta_k^{-1} g_k A^{(k-1)}_{k,\bullet} .$$

The corresponding transformation of right-hand side vector b proceeds as follows. Starting from $b^{(0)} = b' = Pb$, the k-th elimination step, k = 1, ... , n, transforms $b^{(k-1)}$ into $b^{(k)}$ according to

(3.4) $$b^{(k)} = T_k b^{(k-1)} = b^{(k-1)} - \delta_k^{-1} b^{(k-1)}_k g_k.$$

Thus, the given linear system is transformed into the equivalent system $DQ^{-1}x = y$, which is easily solved by calculating

(3.5) $$x = QD^{-1} y .$$

Gauss - Jordan with pivoting

The selection of pivots and the corresponding interchanges take place in a similar way as in Gaussian elimination. In the k-th elimination step, k=1, ..., n, the k-th pivot is selected in the lower right (n-k+1)-th order submatrix of $A^{(k-1)}$, where in principle the same pivoting strategies can be chosen as for Gaussian elimination.

Mostly, partial pivoting is performed, which, as explained above in section 2, can use either row interchanges or column interchanges. The numerical behaviour of the Gauss-Jordan algorithm is quite different, however, for these two strategies, in contrast to the behaviour of Gaussian elimination. The accuracy of the calculated solution is of the same order of magnitude in all cases. The difference manifests itself in the size of the residual, $r = b - A x$, of a calculated approximate solution x.
Gauss-Jordan using partial pivoting with *row* interchanges often yields a much larger residual corresponding to the calculated solution than Gaussian elimination does, as has been shown by Peters & Wilkinson [14]. On the other hand, Gauss-Jordan using partial pivoting with *column* interchanges yields a residual which is mostly not larger than the residual obtained by Gaussian elimination of the same system. This follows from an error analysis by Dekker & Hoffmann [4] which we present below in a slightly modified form.

Error analysis

The main difference between Gaussian elimination and Gauss-Jordan is that the latter algorithm calculates the LU factorization of the matrix and the explicit inverse of matrix U during the reduction of the matrix to diagonal form. The result of the whole algorithm can be described as follows. Let V denote the calculated inverse of U. Then Gauss-Jordan to solve system (1.1) satisfies the following theorem (note that we use notation (1.2) but assume P = I here).

3.6. <u>Theorem.</u> *Gauss-Jordan using partial pivoting with column interchanges, calculating in floating-point arithmetic with machine precision ε, assuming that no arithmetic exceptions occur,* <u>*exactly*</u> *satisfies the following relations, where E_i, for = 1, ..., 4, are appropriate matrices:*

(3.6.1) $A' + E_1 = LU$, $|E_1| \le \varepsilon\, O(n)\, (|A'| + |L|\, |U|)$, *(LU factorization);*

(3.6.2) $(L + E_2)\, y = b$, $|E_2| \le \varepsilon\, O(n)\, |L|$, *(forward substitution);*

(3.6.3) $VU = I + E_3$, $|E_3| \le \varepsilon\, O(n)\, |V|\, |U|$, *(inversion of U);*

(3.6.4) $(V + E_4)\, y = x'$, $| E_4 | \le \varepsilon\, O(n)\, |V|$, *(calculate solution).*

Hence, the calculated solution vector $x = Qx'$ satisfies the <u>*exact*</u> *relation:*

(3.6.5) $(A' + E_1 + E_2U)\, (I + E_3 + E_4U)^{-1}\, x' = b$.

Defining

$$E_5 := E_1 + E_2\, U, \qquad E_6 := E_3 + E_4\, U,$$

we obtain

$$|E_5| < \varepsilon\, O(n)\, (|A'| + |L|\, |U|), \quad |E_6| < \varepsilon\, O(n)\, |V|\, |U|,$$

$$(A' + \delta A')\, x' = (A' + E_5)\, (\, I + E_6)^{-1} x' = b,$$

$$\delta A' = (E_5 - A'\, E_6)\, (I + E_6)^{-1}.$$

Hence, for any vector norm and associated matrix norm, $||\delta A||$ and residual $r := b - A\, x = \delta A\, x$ satisfy, provided $||E6| < 1$:

$$||\delta A|| \le (||E_5|| + ||A||\, ||E_6||) / (1 - ||E_6||),$$

$$||r|| \le ||dA||\, ||x|| \le ||x||\, (||E_5|| + ||A||\, ||E_6||) / (1 - ||E_6||).$$

The first two relations, (3.6.1) and (3.6.2), are the same as for Gaussian elimination; relations (3.6.3) and (3.6.4) are different and reflect the calculation of U^{-1} in the Gauss-Jordan algorithm described. Note that E_2 is lower triangular (as in Gaussian elimination) and that E_3 and E_4 are upper triangular matrices.
A proof of a slightly different theorem, formulated in terms of bounds for the norms of the error matrices E_i, i=1, ..., 4, is given in [4]. That proof is easily adapted to establish the theorem as formulation above. (See also the proof of our theorem in the next section.)

Gauss-Jordan can also be performed with complete pivoting or with monitoring the pivot growth, in order to obtain a more reliable algorithm for very large systems. Obviously, theorem (3.6) also holds for these variants of Gauss-Jordan.

The results of experiments on numerical stability and timing of this algorithm and a similar algorithm for matrix inversion have been published elsewhere [3] [4]. These results show that Gauss-Jordan can be rather efficiently implemented on a supercomputer. Although it requires more work, it is competitive with Gaussian elimination for order n up to nearly 50.

Gauss-Jordan elimination is particularly suitable for calculating the inverse of a matrix. Matrix inversion, by means of Gauss-Jordan or Gaussian elimination, requires about $2n^3$ floating-point operations. Gauss-Jordan matrix inversion can, however, be arranged such that only n^2 vector operations are needed [3]. This cannot be achieved using Gaussian elimination.

4. Gauss-Huard algorithm "méthode des paramètres"

A variant of Gauss-Jordan algorithm introduced by Huard [13], also treated by Cosnard et al. [2], reduces a given matrix to the identity matrix, i.e. yields the same result as Gauss-Jordan, possibly apart from a different diagonal scaling.
Huard's algorithm requires only about $2/3\ n^3$ floating-point operations, the same as needed for Gaussian elimination. The algorithm proceeds in n steps, numbered k = 1, ..., n. Only partial pivoting with column interchanges is possible. Thus, as earlier, we define $A^{(0)} = A' = AQ$ and $b^{(0)} = b' = b$.

As an illustration we display matrix and right-hand side vector at the start of the fourth step. Here a and b denote original elements, and x denotes elements which have been modified in previous steps.

$$A^{(3)} = \begin{pmatrix} 1 & 0 & 0 & x & .. & x \\ 0 & 1 & 0 & x & .. & x \\ 0 & 0 & 1 & x & .. & x \\ a & a & a & a & .. & a \\ : & : & : & : & : & : \\ a & a & a & a & .. & a \end{pmatrix}, \quad b^{(3)} = \begin{pmatrix} x \\ x \\ x \\ b \\ : \\ b \end{pmatrix}.$$

At the start of the k-th step, the first k-1 rows have been transformed such that the upper left submatrix of order k-1 is transformed into the identity matrix. The remaining n-k+1 rows are, however, unchanged at this moment, contrary to what happens in Gaussian elimination and Gauss-Jordan. The k-th elimination step consists of the following three parts.

A) *Row elimination.* The first k-1 elements of the k-th row are eliminated using the first k-1 rows. This requires k-1 vector updates of vectors of length n-k+1, or equivalently, one row vector times matrix subtraction from the k-th row. Hence, 2 (k-1) (n-k+1) floating-point operations are needed.

B) *Pivoting and row scaling.* The k-th pivot is selected in the k-th row obtained in the previous step and columns are interchanged accordingly. Subsequently, the k-th row is scaled such that its diagonal element becomes one; this takes one division and n-k multiplications.

C) *Column elimination.* The elements in the k-th column above the main diagonal are eliminated, where the k-th diagonal element is used as pivot; this final part is quite similar as in Gauss-Jordan elimination; it can be viewed as a rank-one update of the upper-right (k-1) by (n-k) submatrix, which requires 2 (k-1) (n-k) floating-point operations.

Thus, the amount of floating-point operations in the k-th step equals

$$2\ (k-1)\ (n-k+1) + (n-k+1) + 2\ (k-1) * (n-k) = 4\ (k-1)\ (n-k) + n+k-1,$$

so that the total amount of floating-point operations equals

$$2/3\ n^3 - 1/2\ n^2 + 5/6\ n.$$

This is precisely the same as for Gaussian elimination.

Note that the calculated results of Gauss-Jordan and Gauss-Huard, although mathematically the same, mostly are numerically different. The main difference is that in Gauss-Huard matrix L is not explicitly calculated, but only implicitly in product form. This explains the saving in number of operations.

<u>Pivot selection and strategies</u>

In Gauss-Huard algorithm only partial pivoting with *column* interchanges is possible, i.e. in the k-th step after performing part A of the eliminations, the pivot is selected in the k-th *row* and, if needed, a corresponding interchange of columns is performed. Gauss-Huard with this pivoting strategy has about the same numerical behaviour and stability as Gauss-Jordan using partial pivoting with column interchanges, explained above. This follows from the following *new* result.

4.1. <u>*Theorem.*</u> *Gauss-Huard using partial pivoting with column interchanges, calculating in floating-point arithmetic with machine precision ε, assuming that no arithmetic exceptions occur, <u>exactly</u> satisfies the following relations, where E_i, for = 1, ..., 4, are appropriate matrices:*

(4.1.1) $\quad A' + E_1 = LU, \qquad |E_1| \le \varepsilon\ O(n)\ |A'|\ |V|\ |U|;$

(4.1.2) $(L + E_2)\, y = b, \qquad |E_2| \le \varepsilon\, O(n)\, |A'|\, |V|;$

(4.1.3) $VU = I + E_3, \qquad |E_3| \le \varepsilon\, O(n)\, |V|\, |U|;$

(4.1.4) $(V + E_4)\, y = x', \qquad |E_4| \le \varepsilon\, O(n)\, |V|.$

Hence, the calculated solution vector $x = Qx'$ satisfies an <u>exact</u> *relation of the form:*

(4.1.5) $(A' + E_1 + E_2U)\,(I + E_3 + E_4U)^{-1}\, x' = b.$

Note that (4.1.1) and (4.1.2) reflect the effects of row elimination and row scaling of the matrix and the right-hand side vector, and (4.1.3) and (4.1.4) the effects of the column eliminations.

Putting $E_5 = E_1 + E_2\, U$, $E_6 = E_3 + E_4\, U$, we obtain:

$$|E_5| \le \varepsilon\, O(n)\, |A'|\, |V|\, |U|, \qquad |E_6| \le \varepsilon\, O(n)\, |V|\, |U|,$$

$$(A' + \delta A')\, x' = (A' + E_5)\,(\, I + E_6)^{-1}\, x' = b,$$

$$\delta A' = (E_5 - A'\, E_6)\,(I + E_6)^{-1}.$$

Hence, for any vector norm and associated matrix norm, $\|E6\|$ and residual $r := b - A\, x = \delta A\, x$ satisfy, provided $\|E6\| < 1$:

$$\|\delta A\| \le (\|E_5\| + \|A\|\, \|E_6\|) / (1 - \|E_6\|),$$

$$\|r\| \le \|dA\|\, \|x\| \le \|x\|\, (\|E_5\| + \|A\|\, \|E_6\|)/(1 - \|E_6\|).$$

<u>Proof</u>.
We first analyse Huard's method. The pivoting is performed with column interchanges only. Hence, $P = I$ and we write

$$A^{(0)} = A' = AQ, \qquad b^{(0)} = b.$$

As indicated above, the k-th elimination step consists of the following parts:

A) Elimination in k-th row, reducing first k-1 elements to 0, i.e. $A^{(k-1)}$ and similarly $b^{(k-1)}$ are premultiplied by the matrix $I - e_k\, h_k^T$, where

$$h_k^T = (A'_{k,1}\, , \ldots, A'_{k,k-1}\, , 0, \ldots, 0).$$

This yields

$$w_k^T := e_k^T(I - e_k\, h_k^T)\, A^{(k-1)} = e_k^T\, A' - h_k^T\, A^{(k-1)},$$

$$c_k := e_k^T(I - e_k\, h_k^T)\, b^{(k-1)} = b_k - h_k^T\, b^{(k-1)}.$$

B) Pivoting, followed by scaling of k-th row, reducing its k-th element to 1. The row scaling is performed by multiplying the k-th row by the inverse of the k-th pivot $\delta_k = w_{k,k}$.
In other words the whole matrix and the righ-hand side vector are premultiplied by the inverse of

$$D_k := \mathrm{diag}\ (1, \ldots, 1, \delta_k, 1, \ldots, 1).$$

This yields the new k-th row of the matrix and the new k-th element of the right-hand side vector

$$e_k^T\, A^{(k)} = e_k^T\, U = \delta_k^{-1}\, w_k^T \qquad b^{(k)}{}_k = y_k = \delta_k^{-1}\, c_k.$$

C) Elimination in k-th column, reducing k-1 elements to 0, i.e. the k-th column, and similarly the updated right-hand side vector $b^{(k-1)}$, is premultiplied by the matrix $I - v_k e_k^T$, where

$$v_k = (A^{(k-1)}_{1,k} , ..., A^{(k-1)}_{k-1,k} , 0, ..., 0)^T.$$

This latter part is quite similar as in Gauss-Jordan.
Summarizing, the k-th step of Huard's method can be written in the form:

$$(A^{(k)} , b^{(k)}) = (I - v_k e_k^T) D_k^{-1} (I - e_k h_k^T) (A^{(k-1)}, b^{(k-1)}).$$

Let $V^{(k)}$ be the unit upper triangular matrix and $M^{(k)}$ the lower triangular matrix defined by

$$V^{(1)} := I, \qquad V^{(k)} := (I - v_k e_k^T) V^{(k-1)}, \qquad k > 1,$$

$$M^{(1)} := D_1^{-1}, \quad M^{(k)} := D_k^{-1} (I - e_k m_k^T) M^{(k-1)}, \qquad k > 1,$$

where

(4.2) $$m_k := h_k^T V^{(k-1)}.$$

Then we have

$$D_k^{-1} (I - e_k h_k^T) V^{(k-1)} = V^{(k-1)} D_k^{-1} (I - e_k m_k^T).$$

Hence, after k steps we obtain

$$A^{(k)} = V^{(k)} M^{(k)} A', \qquad b^{(k)} = V^{(k)} M^{(k)} b.$$

So, the first k rows of $A^{(k)}$ are equal to those of $V^{(k)} U$, and, similarly, the first k elements of $b^{(k)}$ are equal to those of $V^{(k)} y$, in formula:

(4.3) $$e_i^T A^{(k)} = e_i^T (V^{(k)}U), \qquad i \leq k,$$

(4.4) $$b^{(k)}_i = (V^{(k)}y)_i, \qquad i \leq k.$$

Finally we obtain a product form of the matrix

$$L = (M^{(n)})^{-1} = D_1 (I + e_2 m_2^T) D_2 ... D_{n-1} (I + e_n (m_n^T) D_n$$

and, moreover, $V = V^{(n)}$ as an approximation of U^{-1} and $b^{(n)} = Vy$ as an approximation of x'.

We now are ready to prove the error analysis results stated. We use backward analysis to prove formulas (4.1.1) and (4.1.2) and forward analysis to prove (4.1.3 and (4.1.4).

Proof of (4.1.1).
The row elimination (part A) in the k-th step yields a new k-th row w_k^T whose elements $w_{k,j}$ are bounded by the quantities $s_{k,j}$ as follows:

$$|w_{k,j} | \leq s_{k,j} := |A'_{k,j}| + (|h_k^T| \, |A^{(k-1)}|)_j, \qquad j \geq k.$$

Hence, using well known results of error analysis of inner products [19], the rounding errors in these elements are bounded by $\varepsilon\, k\, s_{k,j}$.
Similarly, the subsequent row scaling in part B yields an additional contribution to the error in these elements bounded by $\varepsilon\, s_{k,j}$.

Hence, the total error of row elimination and row scaling in the k-th step is bounded by

$$\varepsilon\ (k+1)\ s_{k,j} \leq \varepsilon\ O(k)\ (|A'_{k,j}| + (|h_k^T|\ |A^{(k-1)}|)_j), \qquad j \geq k.$$

Adding these contributions together for all elimination steps, and using formula (4.3), we obtain

$$|E_1| \leq \varepsilon\ O(n)\ (|A'_{up}| + |A'_{low}|\ |V^{(k)}U|) \leq \varepsilon\ O(n)\ |A'|\ |V|\ |U|,$$

where A'_{up} denotes the upper triangular part and A'_{low} the lower triangular part of A'.
This proves (4.1.1).

Proof of (4.1.2).
We consider parts A and B of the k-th step operating on the right-hand side vector. Part A calculates the k-th element c_k of $(I - e_k h_k^T)\ b^{(k-1)}$ and part B divides this by the k-th pivot δ_k. The exact calculated value can therefore be written in the form

$$y_k = b^{(k)}_k = \delta_k^{-1}(b_k\ (1 + \varepsilon_k)\ - \Sigma_{i=1}^{k-1}\ (1 + \varepsilon\ O(k))\ (h_k^T\ b^{(k-1)}\ )_i),$$

where $|\varepsilon_k| \leq \varepsilon\ O(k)$. In order to obtain an error formula without error in b_k, we throw the error factor $(1 + \varepsilon_k)$ back to the element $L_{k,k} = \delta_k$, according to backward error analysis, so that we obtain, using (4.4) and (4.2):

$$\Sigma_{i=1}^{k-1}\ (1 + \varepsilon\ O(k))\ (m_k^T\ y)_i) + (1 + \varepsilon_k)^{-1}\ \delta_k\ y_k\ = b_k.$$

Hence,

$$|b_k - (m_k^T\ y + y_k\ \delta_k)| \leq\ \varepsilon\ O(k)\ (|m_k^T| + |\delta_k|\ e_k^T)\ |y|) = \varepsilon\ O(k)\ e_k^T\ |L|\ |y|.$$

Adding these contributions together for all elimination steps, we obtain (4.1.2).

Proof of (4.1.3).
We consider part C (column elimination) of the k-th step. This is a rank-one update yielding a contribution to the error bounded by

$$\varepsilon\ (|A^{(k-1)}_{i,j}| + 2\ |V_{i,k}|\ |U_{k,j}|), \quad i < k,\ j > k.$$

Adding this for all elemination steps yields, using (4.3):

$$|E_3| \leq \varepsilon\ O(n)\ (|A^{(k-1)}| + |V|\ |U|) \leq \varepsilon\ O(n)\ |V|\ |U|,$$

which proves (4.1.3).

Proof of (4.1.4)
We consider part C of the k-th step applied to the right-hand side vector. This is a vector update operation yielding a contribution to the error bounded by

$$\varepsilon\ (|b^{(k-1)}_i| + 2\ |V_{i,k}|\ |y_k|), \qquad i < k.$$

Adding this for all elimination steps and using (4.4), we obtain (4.1.4).
This completes the proof of Theorem 4.1. []

References
[1] P. A. Businger, Monitoring the numerical stability of Gaussian elimination, *Numerische Mathematik* **16** (1971) 360 - 361.
[2] M. Cosnard, Y. Robert & D. Trystram, Résolution parallèle de systèmes linéaires denses par diagonalisation, *E. D. F.- Bulletin de la D. E. R.* **Série C no 2** (1986) 67-88.

[3] T. J. Dekker & W. Hoffmann, Numerical improvement of the Gauss-Jordan algorithm, in A. H. P. van der Burgh & R. M. M. Mattheij, Eds, *Proceedings ICIAM 87, Contributions from the Netherlands* (Paris - La Villette, 1987) 143 - 150.

[4] T. J. Dekker & W. Hoffmann, Rehabilitation of the Gauss-Jordan algorithm, *Numerische Mathematik* **54** (1989) 591-599.

[5] T. J. Dekker, W. Hoffmann & K. Potma, *Parallel algorithms for solving large linear systems* (Technical Report CS-92-12, Department of Computer Systems, University of Amsterdam, August, 1991, to appear in JCAM).

[6] J. J. Dongarra, I. S. Duff, D. C. Sorensen & H. A. van der Vorst, *Solving linear systems on vector and shared memory computers* (SIAM, Philadelphia, 1991).

[7] I. S. Duff, A. M. Erisman & J. K. Reid, *Direct methods for sparse matrices* (Clarendon Press, Oxford, 1986).

[8] G. H. Golub & C. F. Van Loan, *Matrix computations* (second edition, The John Hopkins University Press, Baltimore, 1989).

[9] W. Hoffmann, Solving linear systems on a vector computer, *J. Comp. Appl. Math.* **18** (1987) 353 - 367.

[10] W. Hoffmann, *Basic transformations in linear algebra for vector computing* (Thesis, University of Amsterdam, 1989).

[11] W. Hoffmann, *A fast variant of the Gauss-Jordan algorithm with partial pivoting*, Chapter IV in [15] (1989) 53-60.

[12] W. Hoffmann & K. Potma, *Threshold Pivoting in Gaussian Elimination to Reduce Inter-processor Communication* (Technical Report CS-91-05, Department of Computer Systems, University of Amsterdam, August, 1991).

[13] P. Huard, La méthode Simplex sans inverse explicite, *E. D. F. - Bulletin de la D. E. R.* **Série C no 2** (1979) 79-98.

[14] G. Peters & J. H. Wilkinson, On the stability of Gauss-Jordan elimination with pivoting, *Comm. ACM* **18** (1975) 20 - 24.

[15] K. Potma, *Huard's method for solving linear systems on vector and parallel vector computers* (Technical Report CS-89-12, Department of Computer Systems, University of Amsterdam, 1989).

[16] Y. Robert, *The impact of vector and parallel architectures on the Gaussian elimination algorithm* (Manchester University Press, Manchester, 1990).

[17] L. N. Trefethen & R. S. Schreiber, Average-case Stability of Gaussian Elimination, *SIAM J. Matrix Anal. Appl.* **11** (1990) 335-360.

[18] J. H. Wilkinson, Error analysis of direct methods of matrix inversion, *J. ACM* **8** (1961), 281-330.

[19] J. H. Wilkinson, *Rounding errors in algebraic processes* (Her Majesty's Stationery Office, London, 1963).

Address of the authors:
Department of Computer Systems, University of Amsterdam,
Kruislaan 403, NL-1098 SJ Amsterdam, The Netherlands.
Fax: +31 20 525 7490; e-mail: walter@fwi.uva.nl.