NASA Technical Memorandum 101466
ICOMP-89-2

# On the Equivalence of Gaussian Elimination and Gauss-Jordan Reduction in Solving Linear Equations

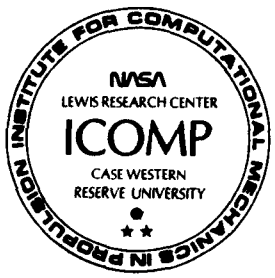


Nai-kuan Tsao
*Wayne State University*
*Detroit, Michigan*

*and Institute for Computational Mechanics in Propulsion*
*Lewis Research Center*
*Cleveland, Ohio*

February 1989

NASA



# On the Equivalence of Gaussian Elimination and Gauss-Jordan Reduction in Solving Linear Equations

Nai-kuan Tsao*
Wayne State University
Detroit, Michigan 48202

and Institute for Computational Mechanics in Propulsion
Lewis Research Center
Cleveland, Ohio 44135

Abstract

A novel general approach to round-off error analysis using the error complexity concepts is described. This is applied to the analysis of the Gaussian Elimination and the Gauss-Jordan scheme for solving linear equations. The results show that the two algorithms are equivalent in terms of our error complexity measures. Thus the inherently parallel Gauss-Jordan scheme can be implemented with confidence if parallel computers are available.

*This work was supported in part by the Space Act Agreement C99066G while the author was visiting ICOMP, NASA Lewis Research Center.

## 1. Introduction

In past decades, the backward error analysis of Wilkinson[3] has enabled us to gain much insight into the behavior of round-off error propagation in numerical algebraic algorithms. One is required, naturally, to possess varying degrees of mathematical sophistication in order to apply his approach to the analysis of numerical algorithms on hand.

The algorithms of Gaussian Elimination (GE) and Gauss-Jordan reduction (GJ) are well known in solving linear equations. The numerical stability of Gaussian Elimination with partial pivoting is shown in [3] , and the stability of Gauss-Jordan reduction is shown in [4] , all using Wilkinson's approach. The results show that in Gaussian Elimination the computed solution $\tilde{x}$ of a given system $Ax = b$ is the exact solution of some neighboring system $(A + E)\tilde{x} = b$ with a reasonable bound for $E$, whereas in Gauss-Jordan reduction the computed solution $\bar{x}$ is such that each component of $\bar{x}$ belongs to the exact solution of a different neighboring system.

In this paper we extend the error complexity concepts in [2] to division operation and apply it to the analysis of the Gauss-Jordan and Gaussian Elimination algorithms to show from an alternative point of view that the two algorithms are indeed equivalent in terms of our error complexity measures. Thus one can implement the inherently parallel Gauss-Jordan reduction algorithm in with confidence for parallel computers. Some preliminary results are presented in Section 2. They are applied in Section 3 to the error complexity analysis of the two algorithms for solving linear system of equations.

## 2. Some Preliminary Results

Consider the simple problem of evaluating the 3 by 3 determinant

$$\det(A) = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}. \tag{2.1}$$

A straightforward algorithm would evaluate (2.1) as

$$\det(A) = a_{11}(a_{22}a_{33} - a_{23}a_{32}) + a_{12}(a_{23}a_{31} - a_{21}a_{33}) + a_{13}(a_{21}a_{32} - a_{22}a_{31}).$$

On the otherhand, one can also express (2.1) as

$$\det(A) = a_{11}a'_{22}a''_{33} \tag{2.2}$$

where

$$a'_{22} = a_{22} - \frac{a_{21}}{a_{11}}a_{12},\ a''_{33} = (a_{33} - \frac{a_{31}}{a_{11}}a_{13}) - \frac{1}{a'_{22}}(a_{32} - \frac{a_{31}}{a_{11}}a_{12})(a_{23} - \frac{a_{21}}{a_{11}}a_{13}).$$

Expanding (2.2), we have

$$\det(A) = a_{11}(a_{22} - \frac{a_{21}}{a_{11}}a_{12})(a_{33} - \frac{a_{31}}{a_{11}}a_{13}) - a_{11}(a_{32} - \frac{a_{31}}{a_{11}}a_{12})(a_{23} - \frac{a_{21}}{a_{11}}a_{13})$$

Which is equivalent to (2.1) once the term

$$a_{11}\frac{a_{21}}{a_{11}}a_{12}\frac{a_{31}}{a_{11}}a_{13}$$

is cancelled out. This is of course unlikely in actual computation due to round-off errors. Thus the latter approach is more likely to incur round-off errors during the computational process.

Our approach in the error analysis of different algorithms designed for the same problem strives to provide such information as the number of round-off error occurrences as well as the extra terms created during the computational process for easy comparison and at the same time enable us to gain more insight into the details of how each algorithm works.

Given a normalized floating-point system with a $t$-digit base $\beta$ mantissa, the following equations can be assumed to facilitate the error analysis of general arithmatic expressions using only $+, -, \times$, or / operations[3]:

$$fl(x\#y) = (x\#y)\Delta,\quad \# \in \{+, -, \times, /\} \tag{2.3}$$

where

$$|\Delta| \le 1 + u, \quad u \le \begin{cases} \frac{1}{2}\beta^{1-t} \text{ for rounded operations} \\ \beta^{1-t} \text{ for chopped operations} \end{cases}$$

and $x$ and $y$ are given machine floating-point numbers and $fl(.)$ is used to denote the computed floating-point result of the given argument. We shall call $\Delta$ the unit $\Delta$ -factor.

In general one can apply (2.3) repeatedly to a sequence of arithmetic steps, and the computed result $z$ can be expressed as

$$z = \frac{z_n}{z_d} \equiv \frac{\sum_{i=1}^{\lambda(z_n)} z_{ni}\Delta^{\sigma(z_{ni})}}{\sum_{j=1}^{\lambda(z_d)} z_{dj}\Delta^{\sigma(z_{dj})}} \tag{2.4}$$

where each $z_{ni}$ or $z_{dj}$ is an exact product of error-free data, and $\Delta^k$ stands for the product of k possibly different $\Delta$-factors. We should emphasize that all common factors between the numerator and denominator should have been factored out before $z$ can be expressed in its final rational form of (2.4). Following [1], we shall henceforth call such an exact product of error-free data a basic term, or simply a term. Thus $\lambda(z_n)$ or $\lambda(z_d)$ is then the total number of such terms whose sum constitutes $z_n$ or $z_d$, respectively, and $\sigma(z_{ni})$ or $\sigma(z_{dj})$ gives the possible number of round-off occurrences during the computational process. We define the following two measures:

maximum error complexity:

$$\sigma(z_n) \equiv \max_{1 \le i \le \lambda(z_n)} \sigma(z_{ni}), \quad \sigma(z_d) \equiv \max_{1 \le j \le \lambda(z_d)} \sigma(z_{dj}) \tag{2.5}$$

cumulative error complexity:

$$s(z_n) \equiv \sum_{i=1}^{\lambda(z_n)} \sigma(z_{ni}), \quad s(z_d) \equiv \sum_{j=1}^{\lambda(z_d)} \sigma(z_{dj}). \tag{2.6}$$

Different algorithms used to compute the same $z$ can then be compared using the above error complexity measures and the number of basic terms created by each algorithm.

For convenience we will use $ch_d(z)$ and $ch_n(z)$ to represent the 3-tuples $\{\lambda(z_d), \sigma(z_d), s(z_d)\}$ and $\{\lambda(z_n), \sigma(z_n), s(z_n)\}$ , respectively, so that the computed $z$ of (2.4) is fully characterized by

$$ch(z) \equiv \frac{ch_n(z)}{ch_d(z)} .$$

The unit $\Delta$-factor is then characterized by

(2.7*a*) $$ch(\Delta) = \{1,1,1\} \equiv \{\Delta\}.$$

One can also obtain easily using (2.5) and (2.6) that

(2.7*b*) $$ch(\Delta^i) = ch^i(\Delta) = \{\Delta\}^i = \{1,i,i\}.$$

In division-free computations any computed $z$ will have only the numerator part $ch_n(z)$. The following lemma is useful in dealing with intermediate computed results:

Lemma 2.1 Given $x$ and $y$ with their associated $ch_n(x)$ and $ch_n(y)$,

(i) if $z = xy$, then

$$\begin{aligned} ch_n(z) &\equiv ch_n(x)ch_n(y) = ch_n(y)ch_n(x) \\ &= \{\lambda(x_n)\lambda(y_n),\ \sigma(x_n) + \sigma(y_n),\ s(x_n)\lambda(y_n) + \lambda(x_n)s(y_n)\}, \end{aligned}$$

(ii) if $z = x \pm y$, then

$$\begin{aligned} ch_n(z) &\equiv ch_n(x) + ch_n(y) = ch_n(y) + ch_n(x) \\ &= \{\lambda(x_n) + \lambda(y_n),\ \max(\sigma(x_n), \sigma(y_n)),\ s(x_n) + s(y_n)\}. \end{aligned}$$

Proof. The results can be obtained easily by expressing $x$ and $y$ as

$$x = \sum_{i=1}^{\lambda(x_n)} x_{ni}\Delta^{\sigma(x_{ni})}, \quad y = \sum_{j=1}^{\lambda(y_n)} y_{nj}\Delta^{\sigma(y_{nj})}$$

and applying (2.5), (2.6) and the definition of $\lambda(z_n)$ to find $ch_n(z)$. Q.E.D.

For general floating-point computations, we have the following lemma:

Lemma 2.2 Given $x$ and $y$ with their associated

$$ch(x) = \frac{ch_n(x)}{ch_d(x)} \equiv \frac{\{\lambda(x_n), \sigma(x_n), s(x_n)\}}{\{\lambda(x_d), \sigma(x_d), s(x_d)\}}, \quad ch(y) = \frac{ch_n(y)}{ch_d(y)} \equiv \frac{\{\lambda(y_n), \sigma(y_n), s(y_n)\}}{\{\lambda(y_d), \sigma(y_d), s(y_d)\}},$$

(i) if $z = fl(x \pm y)$ and there is no common factors between $x_d$ and $y_d$, then

$$ch(z) = \frac{ch_n(z)}{ch_d(z)} = \frac{ch_d(x)ch_n(y)\{\Delta\} + ch_d(y)ch_n(x)\{\Delta\}}{ch_d(x)ch_d(y)}$$

where

$$\begin{aligned}
&\lambda(z_n) = \lambda(x_n)\lambda(y_d) + \lambda(y_n)\lambda(x_d),\\
&\lambda(z_d) = \lambda(x_d)\lambda(y_d),\\
&\sigma(z_n) = 1 + \max(\sigma(x_n) + \sigma(y_d),\ \ \sigma(y_n) + \sigma(x_d)),\\
&\sigma(z_d) = \sigma(x_d) + \sigma(y_d),\\
&s(z_n) = \lambda(x_n)s(y_d) + \lambda(y_d)s(x_n) + \lambda(y_n)s(x_d) + \lambda(x_d)s(y_n) + \lambda(z_n),\\
&s(z_d) = \lambda(x_d)s(y_d) + \lambda(y_d)s(x_d);
\end{aligned}$$

(ii) if $z = fl(x \times y)$ and there is no common factors between $x_n$ and $y_d$ or between $y_n$ and $x_d$, then

$$ch(z) = \frac{ch_n(z)}{ch_d(z)} = \frac{ch_n(x)ch_n(y)\{\Delta\}}{ch_d(x)ch_d(y)}$$

where

$$\lambda(z_n) = \lambda(x_n)\lambda(y_n),$$
$$\lambda(z_d) = \lambda(x_d)\lambda(y_d),$$
$$\sigma(z_n) = 1 + \sigma(x_n) + \sigma(y_n),$$
$$\sigma(z_d) = \sigma(x_d) + \sigma(y_d),$$
$$s(z_n) = \lambda(x_n)s(y_n) + \lambda(y_n)s(x_n) + \lambda(z_n),$$
$$s(z_d) = \lambda(x_d)s(y_d) + \lambda(y_d)s(x_d);$$

(iii) if $z = fl(x/y)$ and there is no common factors between $x_n$ and $y_n$ or between $x_d$ and $y_d$, then

$$ch(z) = \frac{ch_n(z)}{ch_d(z)} = \frac{ch_n(x)ch_d(y)\{\Delta\}}{ch_n(y)ch_d(x)}$$

where

$$\lambda(z_n) = \lambda(x_n)\lambda(y_d),$$
$$\lambda(z_d) = \lambda(x_d)\lambda(y_n),$$
$$\sigma(z_n) = \sigma(x_n) + \sigma(y_d) + 1,$$
$$\sigma(z_d) = \sigma(x_d) + \sigma(y_n),$$
$$s(z_n) = \lambda(x_n)s(y_d) + \lambda(y_d)s(x_n) + \lambda(z_n),$$
$$s(z_d) = \lambda(x_d)s(y_n) + \lambda(y_n)s(x_d).$$

Proof. first we apply (2.3) to each case and obtain

$$z = fl(x\#y) = (x\#y)\Delta, \quad \# \in \{+, -, \times, /\}.$$

The results can then be obtained easily by using Lemmas 2.1 and 2.2. Q.E.D.

## 3. Error Complexity Analysis of the Two Algorithms

Given

$$Ax = b$$

where

$$A = \begin{bmatrix} a_{11}a_{12} & \cdots & a_{1N} \\ a_{21}a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \vdots \\ a_{N1}a_{N2} & \cdots & a_{NN} \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix},$$

it is desired to find $x$. We shall assume that $A, b$ are error-free with $\lambda(a_{ij}) = \lambda(b_i) = 1$ and pivoting is not necessary. For simplicity, the $b$ vector is appended to $A$ as the $(N+1)$-st column of $A$. Thus initially the augmented matrix is such that

$$ch(a_{ij}) = \begin{cases} \bar{c}_1 \text{ if } i = j = 1, \\ c_1 \text{ otherwise .} \end{cases}$$

where

$$\bar{c}_1 = \{\bar{\lambda}_1, \bar{\sigma}_1, \bar{s}_1\} = c_1 = \{\lambda_1, \sigma_1, s_1\} = \{1,0,0\}.$$

The use of $\bar{c}_1$ to distinguish $ch(a_{11})$ from all others is to facilitate the task of identifying common factors as we shall see later. The two algorithms are as follows:

Algorithm GE.

1. {begin Reduction to Triangular Form}
   for $i = 1$ to $N - 1$ do
   for $k = i + 1$ to $N$ do
   $a_{ki} = fl(a_{ki}/a_{ii})$
   for $j = i + 1$ to $N + 1$ do
   $a_{kj} = fl(a_{kj} - a_{ki} \times a_{ij})$
2. {begin Back-Substitution}
   $x_N = fl(a_{N,N+1}/a_{NN})$
   for $i = N - 1$ downto 1 do
   for $j = N$ downto $i + 1$ do
   $a_{i,N+1} = fl(a_{i,N+1} - a_{ij} \times x_j)$
   $x_i = fl(a_{i,N+1}/a_{ii})$

Algorithm GJ.

1. {begin Reduction to Diagonal Form}
   for $i = 1$ to $N$ do
   for $k = 1$ to $N$ (except $i$) do
   $a_{ki} = fl(a_{ki}/a_{ii})$
   for $j = i + 1$ to $N + 1$ do
   $a_{kj} = fl(a_{kj} - a_{ki} \times a_{ij})$
2. {begin Solving Diagonal System}
   for $i = 1$ to $N$ do

$$x_i = fl(a_{i,N+1}/a_{ii})$$

A closer examination of the two algorithms reveals that the computed lower triangular part of the matrix $A$ are identical. The only difference between the two lies in the solution of the upper triangular systems: in Algorithm GE the back-substitution scheme is used, whereas in Algorithm GJ a forward-elimination scheme is used to reduce the upper triangular form to diagonal form for the final solution. Thus error-wise the Algorithm GJ is equivalent to the following modified one:

Algorithm GJm.

1. {begin Reduction to Upper Triangular Form}
   same as in Algorithm GE.
2. {begin Reduction of Upper Triangular to Diagonal Form}
   for $i = 1$ to $N - 1$ do
   for $k = 1$ to $i$ do
   $a_{k,i+1} = fl(a_{k,i+1}/a_{i+1,i+1})$
   for $j = i + 2$ to $N + 1$ do
   $a_{kj} = fl(a_{kj} - a_{k,i+1} \times a_{i+1,j})$
3. {begin Solving Diagonal System}
   same as in Step 2 of Algorithm GJ.

We first give a detailed analysis of Step 1 of both Algorithm GE and Algorithm GJm for $N = 3$. Applying (2.3) to Step 1, we obtain, after the first iteration for $i = 1$ is completed, the following computed results:

$$\begin{matrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a'_{21} & a'_{22} & a'_{23} & a'_{24} \\ a'_{31} & a'_{32} & a'_{33} & a'_{34} \end{matrix}$$

where

$$a'_{k1} = \frac{a_{k1}\Delta}{a_{11}}, \quad a'_{kj} = (a_{kj}\Delta - a'_{k1}a_{1j}\Delta^2) = \frac{a_{kj}a_{11}\Delta - a_{k1}a_{1j}\Delta^3}{a_{11}}, \quad k = 2,3; \quad j = 2,3,4.$$

Applying Lemmas 2.1 and 2.2 to the above equation, we obtain the following matrix for $ch(a_{ij})$:

$$\begin{matrix} \bar{c}_1 & c_1 & c_1 & c_1 \\ \dfrac{c_1\{\Delta\}}{\bar{c}_1} & \dfrac{\bar{c}_2}{\bar{c}_1} & \dfrac{c_2}{\bar{c}_1} & \dfrac{c_2}{\bar{c}_1} \\ \dfrac{c_1\{\Delta\}}{\bar{c}_1} & \dfrac{c_2}{\bar{c}_1} & \dfrac{c_2}{\bar{c}_1} & \dfrac{c_2}{\bar{c}_1} \end{matrix}$$

where

$$c_2 \equiv \{\lambda_2, \sigma_2, s_2\} = \bar{c}_2 \equiv \{\bar{\lambda}_2, \bar{\sigma}_2, \bar{s}_2\} = c_1\bar{c}_1\{\Delta\} + c_1c_1\{\Delta\}^3 = \{1,1,1\} + \{1,3,3\} = \{2,3,4\}.$$

Note since $a_{11}$ is the common denominator of all newly computed results, hence $\bar{c}_1$ is also present as the denominator of all computed items. Again $\bar{c}_2$ is used to distinguish $ch(a'_{22})$ from the rest of $ch(a'_{kj})$.

Similarly after the completion of the second iteration for $i = 2$ of Step 1 in both algorithms, we have the following computed results:

$$\begin{matrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a'_{21} & a'_{22} & a'_{23} & a'_{24} \\ a'_{31} & a''_{32} & a''_{33} & a''_{34} \end{matrix}$$

where

$$a''_{32} = \frac{a'_{32}\Delta}{a'_{22}},$$

$$a''_{3j} = (a'_{3j}\Delta - a''_{32}a'_{2j}\Delta^2), \quad j = 3,4$$

$$= \frac{(a_{33}a_{11}\Delta - a_{31}a_{13}\Delta^3)(a_{22}a_{11}\Delta - a_{21}a_{12}\Delta^3)\Delta - (a_{32}a_{11}\Delta - a_{31}a_{12}\Delta^3)(a_{2j}a_{11}\Delta - a_{21}a_{1j}\Delta^3)\Delta^3}{(a_{22}a_{11}\Delta - a_{21}a_{12}\Delta^3)a_{11}}$$

Note in the above expression, the term $a_{31}a_{13}a_{21}a_{12}$ is not likely to be cancelled out in actual computation as expected in ideal computation. The *ch*-matrix can be obtained from the above expression as

$$\begin{matrix} \bar{c}_1 & c_1 & c_1 & c_1 \\ \dfrac{c_1\{\Delta\}}{\bar{c}_1} & \dfrac{\bar{c}_2}{\bar{c}_1} & \dfrac{c_2}{\bar{c}_1} & \dfrac{c_2}{\bar{c}_1} \\ \dfrac{c_1\{\Delta\}}{\bar{c}_1} & \dfrac{c_2\{\Delta\}}{\bar{c}_2} & \dfrac{\bar{c}_3}{\bar{c}_2\bar{c}_1} & \dfrac{c_3}{\bar{c}_2\bar{c}_1} \end{matrix}$$

where

$$\bar{c}_3 = c_3 = c_2\bar{c}_2\{\Delta\} + c_2c_2\{\Delta\}^3 = \{2,3,4\}^2[\{1,1,1\} + \{1,3,3\}] = \{8,9,48\}.$$

In general we have the following theorem:

Theorem 3.1 The *ch*-matrix of the newly computed part of $A$, after the $i$-th iteration of Step 1 of Algorithm GE is completed, is given as follows:

$$ch(a_{11}) = \bar{c}_1,\ ch(a_{1j}) = c_1,\ \ 2 \le j \le N+1$$

$$ch(a_{ki}) = \frac{c_i\{\Delta\}}{\bar{c}_i},\ \ 1 \le i < k \le N,$$

$$ch(a_{kj}) = \begin{cases} \dfrac{\bar{c}_{i+1}}{\prod_{r=1}^{i}\bar{c}_r} & \text{for } j = k = i+1 \\ \dfrac{c_{i+1}}{\prod_{r=1}^{i}\bar{c}_r} & \text{for other values of } i+1 \le j,k \le N+1 \end{cases}$$

where

$$c_{j+1} = \bar{c}_{j+1} = c_j\bar{c}_j\{\Delta\} + c_jc_j\{\Delta\}^3,\quad 1 \le j \le N-1.$$

Proof. See Appendix I.

Once we have the original system reduced to an upper triangular form, the rest of the steps in Algorithms GE and GJm can be applied. We have the following theorems:

Theorem 3.2 The *ch*-vector of the computed solution $x$ using Algorithm GE is given as

$$ch(x_i) = \frac{c_ic_{i+1}\cdots c_N}{\bar{c}_i\bar{c}_{i+1}\cdots\bar{c}_N}(\{\Delta\} + \{\Delta\}^3)^{N-i}\{\Delta\},\quad 1 \le i \le N.$$

Proof. See Appendix II.

Theorem 3.3 The *ch*-matrix of the newly computed results, after the completion of the $i$-th iteration of Step 2 of Algorithm GJm, is given as:

$$ch(a_{k,i+1}) = \frac{\bar{c}_k(\prod_{r=k}^{i} c_r)(\{\Delta\} + \{\Delta\}^3)^{i-k}\{\Delta\}}{\bar{c}_{i+1}}, \quad 1 \le k \le i,$$

$$ch(a_{kj}) = \frac{(\prod_{r=k}^{i+1} c_r)(\{\Delta\} + \{\Delta\}^3)^{i-k+1}}{\prod_{r=1, r\neq k}^{i+1} \bar{c}_r}, \quad 1 \le k \le i, \quad i+2 \le j \le N+1.$$

The *ch*-vector of the computed solution $x$ is the same as that given in Theorem 3.2.

Proof. See Appendix III.

From Theorem 3.3 we conclude that Algorithm GE and Algorithm GJ are equivalent to each other in terms of our error complexity measures.

## References

[1] V.B. Aggarwal and J.W. Burgmeier, A round-off error model with applications to arithmetic expressions, SIAM J. Computing, 8(1979), pp. 60-72.

[2] N.K. Tsao, A simple approach to the error analysis of division-free numerical algorithms, IEEE Trans. Computers, C-32(1983), pp. 343-351.

[3] J.H. Wilkinson, Rounding Errors in Algebraic Processes, Prentice-Hall, Englewood Cliffs, NJ, 1963.

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

Appendix I.

Proof of Theorem 3.1. We prove by induction on $i$. For $i = 1$ it is true by the results demonstrated for $N = 3$. Assume the truth for $i - 1$. Now for $i$ we have

$$a_{ki}^{(new)} = fl(a_{ki}^{(old)} / a_{ii}^{(old)}) = \frac{a_{ki}^{(old)}\Delta}{a_{ii}^{(old)}}, \quad i < k \le N,$$

By assumption,

$$ch(a_{ki}^{(old)}) = \frac{c_i}{\prod_{r=1}^{i-1}\bar{c}_r}, \quad ch(a_{ii}^{(old)}) = \frac{\bar{c}_i}{\prod_{r=1}^{i-1}\bar{c}_r},$$

hence

$$ch(a_{ki}^{(new)}) = \frac{ch(a_{ki}^{(old)})\{\Delta\}}{ch(a_{ii}^{(old)})} = \frac{c_i\{\Delta\}}{\bar{c}_i}.$$

For $i + 1 \le k \le N$ and $i + 1 \le j \le N + 1$ we have

$$a_{kj}^{(new)} = fl(a_{kj}^{(old)} - a_{ki}^{(new)} \times a_{ij}^{(old)}) = a_{kj}^{(old)}\Delta - a_{ki}^{(new)} \times a_{ij}^{(old)}\Delta^2.$$

Hence

$$ch(a_{kj}^{(new)}) = \frac{c_i}{\prod_{r=1}^{i-1}\bar{c}_r}\{\Delta\} + \frac{c_i\{\Delta\}}{\bar{c}_i}\frac{c_i}{\prod_{r=1}^{i-1}\bar{c}_r}\{\Delta\}^2$$

$$= \frac{c_i\bar{c}_i\{\Delta\} + c_ic_i\{\Delta\}^3}{\prod_{r=1}^{i}\bar{c}_r} \equiv \begin{cases} \dfrac{\bar{c}_{i+1}}{\prod_{r=1}^{i}\bar{c}_r} & \text{if } k = j = i+1, \\ \dfrac{c_{i+1}}{\prod_{r=1}^{i}\bar{c}_r} & \text{otherwise.} \end{cases}$$

This proves the theorem. Q.E.D.

Appendix II.

Proof of Theorem 3.2. We prove by backward induction on $i$. For $i = N$,

$$x_N = fl(a_{N,N+1}/a_{NN}).$$

So that

$$ch(x_N) = \frac{ch(a_{N,N+1})\{\Delta\}}{ch(a_{NN})}.$$

By Theorem 3.1 we have

$$ch(a_{N,N+1}) = \frac{c_N}{\prod_{r=1}^{N-1}\bar{c}_r}, \quad ch(a_{NN}) = \frac{\bar{c}_N}{\prod_{r=1}^{N-1}\bar{c}_r}.$$

Hence

$$ch(x_N) = \frac{c_N\{\Delta\}}{\bar{c}_N}$$

which is true. Now assuming the theorem is true for $x_N, x_{N-1}, \ldots, x_{i+1}$, To obtain $x_i$ we first form

$y_0 = a_{i,N+1}$

for $j = N$ downto $i + 1$ do

$y_{N-j+1} = fl(y_{N-j} - a_{ij} \times x_j)$

and then compute $x_i = fl(y_{N-i}/a_{ii})$. By (2.3) we have

$$y_{N-j+1} = y_{N-j}\Delta - a_{ij}x_j\Delta^2.$$

By using Lemma 2.2 we obtain

$$ch(y_{N-j+1}) = ch(y_{N-j})\{\Delta\} + \frac{c_i}{\prod_{r=1}^{i-1}\bar{c}_r}\frac{c_j c_{j+1}\cdots c_N}{\bar{c}_j\bar{c}_{j+1}\cdots\bar{c}_N}(\{\Delta\} + \{\Delta\}^3)^{N-j}\{\Delta\}^3.$$

It is easily shown that the solution to the above equation is

$$ch(y_{N-j}) = \frac{c_i}{\prod_{r=1}^{i-1}\bar{c}_r}\frac{c_{j+1}\cdots c_N}{\bar{c}_{j+1}\cdots\bar{c}_N}(\{\Delta\} + \{\Delta\}^3)^{N-j}.$$

Therefore

$$ch(y_{N-i}) = \frac{c_i}{\prod_{r=1}^{i-1}\bar{c}_r}\frac{c_{i+1}c_{i+2}\cdots c_N}{\bar{c}_{i+1}\bar{c}_{i+2}\cdots\bar{c}_N}(\{\Delta\} + \{\Delta\}^3)^{N-i}.$$

Now

$$x_i = fl(y_{N-i}/a_{ii}) = \frac{y_{N-i}\Delta}{a_{ii}},$$

hence

$$ch(x_i) = \frac{ch(y_{N-i})\{\Delta\}}{ch(a_{ii})}.$$

Since

$$ch(a_{ii}) = \frac{\bar{c}_i}{\prod_{r=1}^{i-1}\bar{c}_r},$$

the common factor $\prod_{r=1}^{i-1}\bar{c}_r$ can be cancelled and we obtain finally

$$ch(x_i) = \frac{c_i c_{i+1}\cdots c_N}{\bar{c}_i\bar{c}_{i+1}\cdots\bar{c}_N}(\{\Delta\} + \{\Delta\}^3)^{N-i}\{\Delta\}.$$

This proves the theorem. Q.E.D.

Appendix III.

Proof of Theorem 3.3. we prove by induction on $i$. For $i = 1$ we have

$$a_{12}^{(new)} = fl(a_{12}^{(old)}/a_{22}) = \frac{a_{12}^{(old)}\Delta}{a_{22}},$$

hence

$$ch(a_{12}^{(new)}) = \frac{ch(a_{12}^{(old)})\{\Delta\}}{ch(a_{22})} = \frac{c_1\bar{c}_1\{\Delta\}}{\bar{c}_2}.$$

Also

$$a_{1j}^{(new)} = fl(a_{1j}^{(old)} - a_{12}^{(new)} \times a_{2j}), \quad 3 \le j \le N+1.$$

Hence

$$\begin{aligned} ch(a_{1j}^{(new)}) &= ch(a_{1j}^{(old)})\{\Delta\} + ch(a_{12}^{(new)})ch(a_{2j})\{\Delta\}^2 \\ &= c_1\{\Delta\} + \frac{c_1\bar{c}_1}{\bar{c}_2}\{\Delta\}\frac{c_2}{\bar{c}_1}\{\Delta\}^2 \\ &= \frac{c_1\bar{c}_2\{\Delta\} + c_1c_2\{\Delta\}^3}{\bar{c}_2} = \frac{c_1c_2}{\bar{c}_2}(\{\Delta\} + \{\Delta\}^3) \end{aligned}$$

and the theorem is true. Assume now the theorem is true for all $i$ up to $i-1$. For $i$ we have

$$a_{k,i+1}^{(new)} = fl(a_{k,i+1}^{(old)}/a_{i+1,i+1}) = \frac{a_{k,i+1}^{(old)}\Delta}{a_{i+1,i+1}}, \quad 1 \le k \le i.$$

Hence

$$ch(a_{k,i+1}^{(new)}) = \frac{ch(a_{k,i+1}^{(old)})\{\Delta\}}{ch(a_{i+1,i+1})}.$$

By the induction assumption

$$ch(a_{k,i+1}^{(old)}) = \frac{(\prod_{r=k}^{i} c_r)(\{\Delta\} + \{\Delta\}^3)^{i-k}}{\prod_{r=1, r\neq k}^{i} \bar{c}_r}, \quad ch(a_{i+1,i+1}) = \frac{\bar{c}_{i+1}}{\prod_{r=1}^{i} \bar{c}_r}.$$

Hence

$$ch(a_{k,i+1}^{(new)}) = \frac{\bar{c}_k(\prod_{r=k}^{i} c_r)}{\bar{c}_{i+1}} (\{\Delta\} + \{\Delta\}^3)^{i-k}\{\Delta\}, \quad 1 \le k \le i.$$

For

$$a_{kj}^{(new)} = fl(a_{kj}^{(old)} - a_{k,i+1}^{(new)} \times a_{i+1,j}), \quad 1 \le k \le i, \quad i+2 \le j \le N+1,$$

then

$$\begin{aligned} ch(a_{kj}^{(new)}) &= ch(a_{kj}^{(old)})\{\Delta\} + ch(a_{k,i+1}^{(new)})ch(a_{i+1,j})\{\Delta\}^2 \\ &= \frac{(\prod_{r=k}^{i} c_r)(\{\Delta\} + \{\Delta\}^3)^{i-k}\{\Delta\}}{\prod_{r=1, r\neq k}^{i} \bar{c}_r} + \frac{\bar{c}_k(\prod_{r=k}^{i} c_r)}{\bar{c}_{i+1}} (\{\Delta\} + \{\Delta\}^3)^{i-k}\{\Delta\} \frac{c_{i+1}}{\prod_{r=1}^{i} \bar{c}_r}\{\Delta\}^2 \\ &= \frac{[\bar{c}_{i+1}(\prod_{r=k}^{i} c_r)\{\Delta\} + (\prod_{r=k}^{i+1} c_r)\{\Delta\}^3](\{\Delta\} + \{\Delta\}^3)^{i-k}}{\prod_{r=1, r\neq k}^{i+1} \bar{c}_r} \\ &= \frac{(\prod_{r=k}^{i+1} c_r)(\{\Delta\} + \{\Delta\}^3)^{i-k+1}}{\prod_{r=1, r\neq k}^{i+1} \bar{c}_r} \quad \text{since } \bar{c}_{i+1} = c_{i+1}. \end{aligned}$$

This proves the first part of the theorem. For the computed solution $x_i$ we have $x_i = fl(a_{i,N+1}/a_{ii})$, hence

$$ch(x_i) = \frac{ch(a_{i,N+1})\{\Delta\}}{ch(a_{ii})}.$$

Since

$$ch(a_{i,N+1}) = \frac{(\prod_{r=i}^{N} c_r)(\{\Delta\} + \{\Delta\}^3)^{N-i}}{\prod_{r=1, r\neq i}^{N} \bar{c}_r}, \quad ch(a_{ii}) = \frac{\bar{c}_i}{\prod_{r=1}^{i-1} \bar{c}_r},$$

therefore

$$ch(x_i) = \frac{\prod_{r=i}^{N} c_r}{\prod_{r=i}^{N} \bar{c}_r} (\{\Delta\} + \{\Delta\}^3)^{N-i}\{\Delta\}.$$

This proves the theorem. Q.E.D.

NASA
National Aeronautics and Space Administration

# Report Documentation Page

| 1. Report No. | 2. Government Accession No. | 3. Recipient's Catalog No. |
|---|---|---|
| NASA TM-101466<br>ICOMP-89-2 | | |

| 4. Title and Subtitle | 5. Report Date |
|---|---|
| On the Equivalence of Gaussian Elimination and Gauss-Jordan Reduction in Solving Linear Equations | February 1989 |
| | 6. Performing Organization Code |
| 7. Author(s) | 8. Performing Organization Report No. |
| Nai-kuan Tsao | E-4577 |
| | 10. Work Unit No. |
| | 505-62-21 |
| 9. Performing Organization Name and Address | 11. Contract or Grant No. |
| National Aeronautics and Space Administration<br>Lewis Research Center<br>Cleveland, Ohio 44135-3191 | |
| | 13. Type of Report and Period Covered |
| | Technical Memorandum |
| 12. Sponsoring Agency Name and Address | 14. Sponsoring Agency Code |
| National Aeronautics and Space Administration<br>Washington, D.C. 20546-0001 | |

15. Supplementary Notes

Nai-kuan Tsao, Wayne State University, Detroit, Michigan 48202. This work was supported in part by the Institute for Computational Mechanics in Propulsion, NASA Lewis Research Center (work funded under Space Act Agreement C99066G).

16. Abstract

A novel general approach to round-off error analysis using the error complexity concepts is described. This is applied to the analysis of the Gaussian Elimination and the Gauss-Jordan scheme for solving linear equations. The results show that the two algorithms are equivalent in terms of our error complexity measures. Thus the inherently parallel Gauss-Jordan scheme can be implemented with confidence if parallel computers are available.

| 17. Key Words (Suggested by Author(s)) | 18. Distribution Statement |
|---|---|
| Error analysis; Parallel algorithms; Gaussian elimination; Gauss-Jordan reduction; Linear equations | Unclassified – Unlimited<br>Subject Category 64 |

| 19. Security Classif. (of this report) | 20. Security Classif. (of this page) | 21. No of pages | 22. Price* |
|---|---|---|---|
| Unclassified | Unclassified | 22 | A03 |



*For sale by the National Technical Information Service, Springfield, Virginia 22161