NASA Technical Memorandum 102036
ICOMP-89-11

On the Equivalence of a Class of Inverse Decomposition Algorithms for Solving Systems of Linear Equations

Nai-kuan Tsao
Wayne State University
Detroit, Michigan

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

May 1989

On the Equivalence of a Class of Inverse Decomposition Algorithms for Solving Systems of Linear Equations

Nai-kuan Tsao*
Wayne State University
Detroit, Michigan

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

Summary

A class of direct inverse decomposition algorithms for solving system of linear equations is presented. Their behavior in the presence of round-off errors is analyzed. It is shown that under some mild restrictions on their implementation, the class of direct inverse decomposition algorithms presented are equivalent in terms of our error complexity measures.

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

1. Introduction

Given a system of linear equations

$$Ax = b,$$

where

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

the solution vector x can be found as

$$x = A^{-1}b$$

provided A is non-singular. Using direct methods the A^{-1} is usually decomposed as a product of elementary matrices which reduces the original A to an identity matrix. This is true whether the method selected is of the Gaussian Elimination type which triangularize A or the Gauss-Jordan type in which diagonalization of A takes place. In actual computation the decomposed A^{-1} is of course only an approximation to the exact one due to round-off errors incurred during the execution of the computational steps of the specific method chosen. Since there are many variations in this class of decomposition methods, one wonders whether one variation is "better" than the other in terms of accuracy of the computed solution.

In this paper we show, under mild restrictions on the implementation of some algorithms, that all decomposition methods are "equivalent" in terms of our error complexity measures. Some preliminary results are presented in Section 2. The classification and analysis of decomposition methods are given in Section 3.

2. Some Preliminary Results

Given a normalized floating-point system with a t-digit base β mantissa, the following equations can be assumed to facilitate the error analysis of general arithmatic expressions using only $+, -, *$, or $/$ operations[1]:

(2.1) $$fl(x\#y) = (x\#y)\Delta, \quad \# \in \{+, -, *, /\}$$

where

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

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 Δ the unit Δ -factor.

In general one can apply (2.1) repeatedly to a sequence of arithmetic steps, and the computed result z can be expressed as

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

where each z_{ni} or z_{dj} is an exact product of error-free data, and Δ^k stands for the product of k possibly different Δ-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 (2.2). Following [2], 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. As an example, consider the solution of two linear equations in two variables as follows:

$$ax_1 + bx_2 = c$$
$$dx_1 + ex_2 = f$$

The exact solution x_2 can be found as

$$x_2^* = \frac{f - \frac{d}{a} * c}{e - \frac{d}{a} * b}$$

The computed x_2 is the result of the following expression:

$$x_2 = fl(w_1/w_2),\ w_1 = fl(f - f_1),\ w_2 = fl(e - e_1),$$

$$f_1 = fl(d_1 * c),\ e_1 = fl(d_1 * b),\ d_1 = fl(d/a).$$

Applying (2.1) repeatedly to the above expression, we have

$$x_2 = \frac{f\Delta - \frac{dc}{a}\Delta^3}{e\Delta - \frac{db}{a}\Delta^3} = \frac{af\Delta - dc\Delta^3}{ae\Delta - db\Delta^3}$$

which is of the form of (2.2). We define the following two measures:

<u>maximum error complexity:</u>

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

<u>cumulative error complexity:</u>

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

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 $\bar{z}_d$ and $\bar{z}_n$ 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 (2.2) is fully characterized by

$$\bar{z} \equiv \frac{\bar{z}_n}{\bar{z}_d}.$$

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

<u>Lemma 2.1</u> Given x and y with their associated $\bar{x}_n$ and $\bar{y}_n$,

(i) if $z = xy$, then

$$\bar{z}_n \equiv \bar{x}_n\bar{y}_n = \bar{y}_n\bar{x}_n = \{\lambda(x_n)\lambda(y_n),\ \sigma(x_n) + \sigma(y_n),\ s(x_n)\lambda(y_n) + \lambda(x_n)s(y_n)\},$$

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

$$\bar{z}_n \equiv \bar{x}_n + \bar{y}_n = \bar{y}_n + \bar{x}_n = \{\lambda(x_n) + \lambda(y_n),\ \max(\sigma(x_n), \sigma(y_n)),\ s(x_n) + s(y_n)\}.$$

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 (2.3), (2.4) and the definition of $\lambda(z_n)$ to find $\bar{z}_n$. Q.E.D.

The unit Δ-factor is then defined as

(2.5a) $$\bar{\Delta} \equiv \{1,1,1\}.$$

One can obtain easily using Lemma 2.1 to get

(2.5b) $$\bar{\Delta}^i = \{1,i,i\}.$$

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

Lemma 2.2 Given x and y with their associated

$$\bar{x} = \frac{\bar{x}_n}{\bar{x}_d} \equiv \frac{\{\lambda(x_n), \sigma(x_n), s(x_n)\}}{\{\lambda(x_d), \sigma(x_d), s(x_d)\}}, \quad \bar{y} = \frac{\bar{y}_n}{\bar{y}_d} \equiv \frac{\{\lambda(y_n), \sigma(y_n), s(y_n)\}}{\{\lambda(y_d), \sigma(y_d), s(y_d)\}},$$

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

$$\bar{z} = \frac{\bar{z}_n}{\bar{z}_d} = \frac{\bar{x}_d\bar{y}_n\bar{\Delta} + \bar{y}_d\bar{x}_n\bar{\Delta}}{\bar{x}_d\bar{y}_d}$$

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

$$\bar{z} = \frac{\bar{z}_n}{\bar{z}_d} = \frac{\bar{x}_n\bar{y}_n\bar{\Delta}}{\bar{x}_d\bar{y}_d}$$

where

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

(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

$$\bar{z} = \frac{\bar{z}_n}{\bar{z}_d} = \frac{\bar{x}_n \bar{y}_d \bar{\Delta}}{\bar{y}_n \bar{x}_d}$$

where

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

Proof. First we apply (2.1) to each case and obtain

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

The results can then be obtained easily by using (2.3) and (2.4). Q.E.D.

Often one needs to add up an extended sum given by

$$\hat{z} = \sum_{i=1}^{k} x_i = \frac{\hat{z}_n}{\hat{z}_d} = \frac{1}{\hat{z}_d} \sum_{i=1}^{k} \hat{z}_{ni}. \tag{2.6}$$

The order of summation certainly has to be specified. One would like to select an order to minimize the incurred additional error complexities of the final computed z. We consider the following four strategies.

If the items are added recursively in parallel by divide-and- conquer, then the strategy is called left-heavy if

$$\hat{z} = \hat{z}_1 + \hat{z}_2$$

where

$$\hat{z}_1 = \sum_{i=1}^{\lceil k/2 \rceil} x_i, \quad \hat{z}_2 = \sum_{i=\lceil k/2 \rceil + 1}^{k} x_i$$

Similarly the strategy is called right-heavy if

$$\hat{z} = \hat{z}_3 + \hat{z}_4, \quad \hat{z}_3 = \sum_{i=1}^{\lfloor k/2 \rfloor} x_i, \quad \hat{z}_4 = \sum_{i=\lfloor k/2 \rfloor + 1}^{k} x_i.$$

If the items are summed up in sequential order, then we have the common strategies of left-to-right or right-to-left. We have the following useful lemma:

Lemma 2.3 Given (2.6) and it is desired to find

$$z = fl(\sum_{i=1}^{k} x_i) = \frac{z_n}{\hat{z}_d}$$

where

$$z_n = fl(\sum_{i=1}^{k} \hat{z}_{ni}),$$

$$2^{k-2}\lambda(\hat{z}_{n1}) = 2^{k-2}\lambda(\hat{z}_{n2}) = 2^{k-3}\lambda(\hat{z}_{n3}) = \dots = 2^{0}\lambda(\hat{z}_{nk}),$$

$$\sigma(\hat{z}_{n1}) + k - 1 \le \sigma(\hat{z}_{n2}) + k - 1 \le \sigma(\hat{z}_{n3}) + k - 2 \le \dots \le \sigma(\hat{z}_{nk}) + 1,$$

then

$$(i)\ \lambda(z_n) = \lambda(\hat{z}_n) = \sum_{i=1}^{k} \lambda(\hat{z}_{ni}) \text{ regardless of the strategy chosen,}$$

$$(ii)\ \sigma(z_n) = \sigma(\hat{z}_{nk}) + w \quad \text{where} \quad w = \begin{cases} \lceil \log k \rceil & \text{if the strategy is right-heavy,} \\ \lfloor \log k \rfloor & \text{if the strategy is left-heavy,} \\ k-1 & \text{if the strategy is right-to-left,} \\ 1 & \text{if the strategy is left-to-right.} \end{cases}$$

$$(iii)\ s(z_n) = \sum_{i=1}^{k} s(\hat{z}_{ni}) + \begin{cases} (2^k - 2)\lambda(\hat{z}_{n1}) & \text{if the strategy is left-to-right,} \\ [1 + (2k-3)2^{k-2}]\lambda(\hat{z}_{n1}) & \text{if the strategy is right-to-left,} \end{cases}$$

$$(iv)\ \sum_{i=1}^{k} s(\hat{z}_{ni}) + \lfloor \log k \rfloor 2^{k-1}\lambda(\hat{z}_{n1}) \le s(z_n) \le \sum_{i=1}^{k} s(\hat{z}_{ni}) + \lceil \log k \rceil 2^{k-1}\lambda(\hat{z}_{n1})$$

if the strategy is either left-heavy or right-heavy.

Proof. The results for $\lambda(z_n)$ are obvious. For error complexities consider first the sequential strategies. If the strategy is left-to-right , then we can easily obtain

$$z_n = \hat{z}_{n1}\Delta^{k-1} + \hat{z}_{n2}\Delta^{k-1} + \dots + \hat{z}_{nk}\Delta^{1}.$$

Applying Lemma 2.1 to the above equation, we obtain

$$\begin{aligned} \sigma(z_n) &= \max(\sigma(\hat{z}_{n1}) + k - 1,\ \sigma(\hat{z}_{n2}) + k - 1,\ \dots,\ \sigma(\hat{z}_{nk}) + 1) \\ &= \sigma(\hat{z}_{nk}) + 1 \end{aligned}$$

by assumption. Also

$$\begin{aligned} s(z_n) &= \sum_{i=1}^{k} s(\hat{z}_{ni}) + (k-1)\lambda(\hat{z}_{n1}) + (k-1)\lambda(\hat{z}_{n2}) + \dots + (1)\lambda(\hat{z}_{nk}) \\ &= \sum_{i=1}^{k} s(\hat{z}_{ni}) + (2^k - 2)\lambda(\hat{z}_{n1}) \end{aligned}$$

by simplification. Hence the theorem is true.

If the strategy is right-to-left, then we have

$$z_n = \hat{z}_{n1}\Delta^1 + \ \dots\ + \hat{z}_{n,k-1}\Delta^{k-1} + \hat{z}_{nk}\Delta^{k-1}$$

By repeated application of Lemma 2.1 we have certainly

$$\sigma(z_n) = \sigma(\hat{z}_{nk}) + k - 1.$$

Also

$$s(z_n) = \sum_{i=1}^{k} s(\hat{z}_{ni}) + \lambda(\hat{z}_{n1}) + 2\lambda(\hat{z}_{n2}) + \dots + (k-1)[\lambda(\hat{z}_{n,k-1}) + \lambda(\hat{z}_{nk})]$$

which can also be simplified to the desired form. Hence the theorem is true in this case. For the parallel strategies, the computed z_n can be expressed as

$$z_n = \hat{z}_{n1}\Delta^{j_1} + \hat{z}_{n2}\Delta^{j_2} + \dots + \hat{z}_{nk}\Delta^{j_k}$$

where

$$j_1 = \lceil \log k \rceil \geq j_2 \geq \dots \geq j_k = \lfloor \log k \rfloor \text{ if the strategy is left-heavy,}$$
$$j_1 = \lfloor \log k \rfloor \leq j_2 \leq \dots \leq j_k = \lceil \log k \rceil \text{ if the strategy is right-heavy,}$$

Now if the strategy is right-heavy, it is obvious that

$$\sigma(z_n) = \sigma(\hat{z}_{nk}) + \lceil \log k \rceil.$$

If the strategy is left-heavy, then by assumption we have

$$\sigma(\hat{z}_{nk}) + \lfloor \log k \rfloor \geq \sigma(\hat{z}_{ni}) + \lfloor \log k \rfloor + k - i \geq \sigma(\hat{z}_{ni}) + \lceil \log k \rceil \quad \text{for } i < k$$

Hence

$$\sigma(z_n) = \sigma(\hat{z}_{nk}) + \lfloor \log k \rfloor.$$

The cumulative error complexity results are obvious. This completes our proof. Q.E.D.

Now from the results of Lemma 2.3 we see obviously that $\sigma(z_n)$ will be minimal if the strategy is left-to-right. For $s(z_n)$ we see that

$$2^k - 2 \leq \lceil \log k \rceil 2^{k-1} \leq [1 + (2k-3)2^{k-2}], \text{ for } k > 3 \text{ or } k = 2.$$

Furthermore for $k = 3$ then we have

$$z_n = \begin{cases} \hat{z}_{n1}\Delta^2 + \hat{z}_{n2}\Delta^2 + \hat{z}_{n3}\Delta & \text{if the strategy is left-heavy,} \\ \hat{z}_{n1}\Delta + \hat{z}_{n2}\Delta^2 + \hat{z}_{n3}\Delta^2 & \text{if the strategy is right-heavy.} \end{cases}$$

Hence

$$s(z_n) = s(\hat{z}_{n1}) + s(\hat{z}_{n2}) + s(\hat{z}_{n3}) + \begin{cases} 6\lambda(\hat{z}_{n1}) \text{ if the strategy is left-heavy,} \\ 7\lambda(\hat{z}_{n1}) \text{ if the strategy is right-heavy,} \\ 6\lambda(\hat{z}_{n1}) \text{ if the strategy is left-to-right,} \\ 7\lambda(\hat{z}_{n1}) \text{ if the strategy is right-to-left.} \end{cases}$$

Thus the left-to-right strategy gives us, in all cases, the minimal cumulative error complexity also in the computed result. We conclude with the following theorem:

Theorem 2.1 If (2.6) is to be computed and the conditions specified in Lemma 2.3 are satisfied, then one should choose the strategy left-to-right in order to minimize both the maximum and cumulative error complexity of the computed extended sum.

3. Inverse Decomposition Methods

To find the inverse decomposition, one applies a sequence of elementary trasformations to the matrix A so that zeroes are created in the lower and upper parts of the matrix. The final reduced matrix is of course the identity matrix so that an implicit A^{-1} can be obtained as a product of the sequence of elementary matrices which can then be applied to b to obtain the desired solution $x = A^{-1}b$. In general a large collection of methods can be classified as the usual triangular decomposition of A into a product of lower and upper triangular matrices followed by explicitly or implicitly inverting the calculated triangular matrices. To see this consider the diagonalization of an $N \times N$ matrix. We shall assume that pivoting is not necessary. The Gauss-Jordan method would create a sequence of matrices $C_1, C_2, \ldots$ and C_N such that

$$C_N \ldots C_2 C_1 A = D\,, \quad C_i = I_N - \sum_{j=1, j\neq i}^{N} c_{ji} e_j e_i^T$$

where I_N and e_i^T represent the $N \times N$ identity matrix and the transpose of the i-th column of the identity matrix , respectively. The matrix C_i is chosen such that the transformed matrix $C_i C_{i-1} \ldots C_1 A$ will have all its off-diagonal elements of the first i columns zeroed. It is also obvious that the matrix C_i depends on the previously generated $C_{i-1}, C_{i-2}, \ldots, C_1$ and A. Now C_i can be expressed as

$$C_i = L_i U_i = U_i L_i \,, \quad L_i = I_N - \sum_{j=i+1}^{N} c_{ji} e_j e_i^T \,, \quad U_i = I_N - \sum_{j=1}^{i-1} c_{ji} e_j e_i^T$$

where U_i and L_i represent, respectively, the upper and lower part of C_i. Furthermore we can easily show by induction that

$$C_N \dots C_2 C_1 = U_N \dots U_3 U_2 L_{N-1} \dots L_2 L_1$$

Hence

$$U^{-1} L^{-1} A = D \,, \quad U^{-1} = U_N \dots U_3 U_2 \,, \quad L^{-1} = L_{N-1} \dots L_2 L_1 .$$

Since the steps in $L^{-1}A$ describe the elimination stage of the Gaussian Elimination method in converting A to an upper triangular form and the subsequent $U^{-1}(L^{-1}A)$ is the result of a forward elimination process to create zeroes in the upper triangular $L^{-1}A$ by columns, the Gauss-Jordan method is therefore numerically equivalent to the elimination stage of Gaussian Elimination method followed by a forward-elimination process to diagonalize the intermediate upper triangular form. More precisely, the Gaussian Elimination method explicitly finds a matrix L whose inverse L^{-1}, in implicit product form, is used to reduce the matrix A to an upper triangular matrix U such that

$$A = LU \,, \quad L^{-1} A = U \,.$$

The Gauss-Jordan method creates the same L and an explicit unit-diagonal upper triangular matrix M such that

$$M L^{-1} A = D.$$

Henceforth we shall restrict our attention to the class of methods by which the matrix A is decomposed first into a product of lower and upper triangular matrix in one of the following two forms:

(3.1a) $$A = LU$$

$$A = PR \tag{3.1b}$$

where L and R are unit-diagonal lower and upper triangular matrix, respectvely; and P and U are general lower and upper triangular matrix, respectively. Once the matrix A is decomposed, then L, U, P, and R can be explicitly or implicitly inverted so that A^{-1} can be obtained.

Expanding the products in (3.1a) and (3.1b) explicitly, the decompositions can be obtained by the following recursive equations:

$$\begin{aligned} u_{kj} &= fl(a_{kj} - \sum_{t=1}^{k-1} l_{kt}u_{tj}), \quad 1 \le k \le j \le N, \\ l_{ji} &= fl((a_{ji} - \sum_{t=1}^{i-1} l_{jt}u_{ti})/u_{ii}), \quad 1 \le i < j \le N, \end{aligned} \tag{3.1a1}$$

$$\begin{aligned} p_{jk} &= fl(a_{jk} - \sum_{t=1}^{k-1} p_{jt}r_{tk}), \quad 1 \le k \le j \le N, \\ r_{ij} &= fl((a_{ij} - \sum_{t=1}^{i-1} p_{it}r_{tj})/p_{ii}), \quad 1 \le i < j \le N. \end{aligned} \tag{3.1b1}$$

The equations in (3.1a1) simply say that any u_{kj} can be computed as soon as the k-th row of L and the first k-1 elements of the j-th column of U are available, and any l_{ji} can be computed as soon as the first $i-1$ elements of the j-th row of L and the i -th column of U are available. Equation (3.1b1) can be interpreted similarly. The only freedom left to an algorithm designer is the order in which the summations in (3.1a1) and (3.1b1) should be excuted. We shall call an order optimal if the computed result has the minimal maximum and cumulative error complexities. We have the following theorem:

Theorem 3.1 Let the given matrix A be such that

$$\bar{a}_{11} = c_{1*} = \{1,0,0\}, \quad \bar{a}_{ij} = c_1 = \{1,0,0\}, \quad (i,j) \neq (1,1).$$

The optimal order to compute the summations in (3.1a1) and (3.1b1) is the left-to-right strategy for their evaluation using the explicit expressions given by

(3.1a2)
$$u_{kj} = fl(a_{kj} - l_{k1}u_{1j} - l_{k2}g_{2j} - \dots - l_{k,k-1}u_{k-1,j}), \quad 1 \le k \le j \le N,$$
$$l_{ji} = fl((a_{ji} - l_{j1}u_{1i} - l_{j2}u_{2i} - \dots - l_{j,i-1}u_{i-1,i})/u_{ii}), \quad 1 \le i < j \le N$$

(3.1b2)
$$p_{jk} = fl(a_{jk} - p_{j1}r_{1k} - p_{j2}r_{2k} - \dots - p_{j,k-1}r_{k-1,k}), \quad 1 \le k \le j \le N,$$
$$r_{ij} = fl(a_{ij} - p_{i1}r_{1j} - p_{i2}r_{2j} - \dots - p_{i,i-1}r_{i-1,j}), \quad 1 \le i < j \le N$$

and the computed u_{kj}, l_{ji}, p_{jk}, and r_{ij} satisfy

$$\bar{u}_{kk} = \bar{p}_{kk} = \frac{c_{k*}}{c_{1*}c_{2*} \dots c_{k-1*}}, \quad \bar{u}_{kj} = \bar{p}_{jk} = \frac{c_k}{c_{1*}c_{2*} \dots c_{k-1*}}, \quad k+1 \le j \le N,$$
$$\bar{l}_{ji} = \bar{r}_{ij} = \frac{c_i\bar{\Delta}}{c_{i*}}, \quad 1 \le i < j \le N$$

where

$$c_{k+1*} = \{\lambda(c_{k+1*}), \sigma(c_{k+1*}), s(c_{k+1*})\} = c_{k+1} = \{\lambda(c_{k+1}), \sigma(c_{k+1}), s(c_{k+1})\} = c_k c_k \bar{\Delta}_{13},$$
$$\bar{\Delta}_{13} \equiv \bar{\Delta} + \bar{\Delta}^3, \quad 1 \le k \le N-1.$$

Proof. See Appendix I.

By Theorem 3.1 we see that all variations of the class of methods for decomposing A into LU or PR are equivalent among those methods of their respective class in terms of our complexity measures as long as the left-to-right strategy is adhered to in the evaluation of (3.1a2) or (3.1b2). Two variations each for the decompositon of A into forms given by (3.1a) and (3.1b) are listed below:

Algorithm G {Gaussian elimination for $A = LU$}

```
for $i = 1$ to $N-1$ do
  for $j = i+1$ to $N$ do
    $l_{ji} = fl(a_{ji}/a_{ii})$
    for $k = i+1$ to $N$ do
      $a_{jk} = fl(a_{jk} - l_{ji} \times a_{ik})$
for $k = 1$ to $N$ do
  for $j = k$ to $N$ do
    $u_{kj} = a_{kj}$
```

Algorithm DL {Doolittle method for $A = LU$}

```
for $j = 1$ to $N$ do
  for $i = 1$ to $j - 1$ do
    $l_{ji} = fl((a_{ji} - \sum_{k=1}^{i-1} l_{jk}u_{ki})/u_{ii})$
  for $k = j$ to $N$ do
    $u_{jk} = fl(a_{jk} - \sum_{i=1}^{j-1} l_{ji}g_{ik})$
```

Algorithm G1 {Gaussian Elimination for $A = PR$}

```
for $i = 1$ to $N - 1$ do
  for $k = i + 1$ to $N$ do
    $r_{ik} = fl(a_{ik}/a_{ii})$
    for $j = i + 1$ to $N$ do
      $a_{jk} = fl(a_{jk} - a_{ji} \times r_{ik})$
for $k = 1$ to $N$ do
  for $j = k$ to $N$ do
    $p_{jk} = a_{jk}$
```

Algorithm CR {Crout method for $A = PR$}

```
for $k = 1$ to $N$ do
  for $j = 1$ to $k$ do
    $p_{kj} = fl(a_{kj} - \sum_{m=1}^{j-1} p_{km}r_{mj})$
  for $j = k + 1$ to $N$ do
    $r_{kj} = fl((a_{kj} - \sum_{m=1}^{k-1} p_{km}r_{mj})/p_{kk})$
```

We are now ready to find the inverses of L, U, P, R either implicitly or explicitly. Given L, the product form of L^{-1} can be obtained without additional computation as

$$L_G^{-1} = L_{N-1,c}^{-1} \cdots L_{1c}^{-1}, \quad L_{ic}^{-1} = I_N - \sum_{j=i+1}^{N} l_{ji}e_je_i^T, \quad 1 \le i \le N-1, \tag{3.2a}$$

$$L_{DL}^{-1} = L_{N-1,r}^{-1} \cdots L_{2r}^{-1}, \quad L_{ir}^{-1} = I_N - \sum_{j=1}^{i-1} l_{ij}e_ie_j^T, \quad 2 \le i \le N. \tag{3.2b}$$

On the other hand, one can also obtain explicitly a lower unit-diagonal matrix M such that $ML = I_N$ and

$$L_{Mc}^{-1} = M_{1c} \cdots M_{N-1,c}, \quad M_{ic} = I_N - \sum_{j=i+1}^{N} m_{ji}e_je_i^T, \quad 1 \le i \le N-1, \tag{3.2c}$$

(3.2d) $$L_{Mr}^{-1} = M_{2r} \dots M_{Nr}, \quad M_{ir} = I_N - \sum_{j=1}^{i-1} m_{ij} e_i e_j^T, \quad 2 \le i \le N.$$

Note that (3.2a), (3.2b), (3.2c), and (3.2d) can be regarded as methods creating zeroes in the given L by column-wise forward , row-wise forward, column-wise backward, and row-wise backward elimination, respectively.

Solving for $ML = I_N$ we find that

$$m_{i+k,i} = fl(\, l_{i+k,i} - \sum_{j=1}^{k-1} m_{i+k,i+j} l_{i+j,i}), \quad 1 \le k \le N-1, \quad 1 \le i \le N-k.$$

In other words, $m_{i+k,i}$ depends on those elements of the $(i+k)$ -th row of M to its right, as well as $l_{i+k,i}$ and those above it in the i-th column of L. A proper algorithm is the following:

Algorithm M

```
for $k = 1$ to $N-1$ do
  for $i = 1$ to $N-k$ do
    $j = i + k$
    $m_{ji} = fl(\, l_{ji} - m_{j,j-1} l_{j-1,i} - \dots - m_{j,i+1} l_{i+1,i})$
```

We have the following theorem:

Theorem 3.2a Given L whose error complexities satisfy Theorem 3.1, then the optimal order to find M using Algorithm M is again the left-to-right strategy for the summations. The computed M has error complexities given as

$$\overline{m}_{i+k,i} = (\prod_{j=1}^{k} c_{i+j-1})\overline{\Delta}_{13}^{k-1}\overline{\Delta}/(\prod_{j=1}^{k} c_{i+j-1*}).$$

Proof. See Appendix II.

Since L^{-1} is applied to A and $L^{-1}A = U$, the same operations need to be applied to the right hand side vector b in solving $Ax = b$ so that the reduced system $Ux = f$ can be solved later where $f = L^{-1}b$. One natural question is whether (3.2a) through (3.2d) are equivalent in the sense that the computed f using any of them will have the same error complexities. For (3.2a) and (3.2b) this is true as we can simply augment A with b as its $(N + 1)$-st column and a decomposition of $A = LU$ can be performed with U being a trapezoidal matrix having f as its last column. For (3.2c) and (3.2d) the results of Theorem 3.2a can be applied and we can easily obtain the following theorem by induction:

Theorem 3.2b If the given right hand side vector b is such that $\bar{b}_i = \{1,0,0\}$, then the computed

$$f = fl(L^{-1}b)$$

using any one of (3.2a) through (3.2d) is such that

$$\bar{f}_i = \bar{u}_{iN}, \quad 1 \leq i \leq N$$

provided the summations used to evaluate f_i in (3.2b) and (3.2d) are carried out using the left-to-right strategy in the following expressions:

(3.2b1) $$f_i = fl(\, b_i - l_{i1}f_1 - l_{i2}f_2 - \dots - l_{i,i-1}f_{i-1}\,), \quad 1 \leq i \leq N-1,$$

(3.2d1) $$f_i = fl(\, b_i - m_{i,i-1}b_{i-1} - m_{i,i-2}b_{i-2} - \dots - m_{i1}b_1\,), \quad 1 \leq i \leq N-1.$$

To find P^{-1} the situation is similar. Two different forms of P^{-1} can be found without additional effort:

(3.3a) $$P_{G1}^{-1} = D_N^{-1}P_{N-1,c}^{-1} \dots D_2^{-1}P_{1c}^{-1}D_1^{-1},$$

(3.3b) $$P_{CR}^{-1} = D_N^{-1}P_{Nr}^{-1} \dots D_2^{-1}P_{2r}^{-1}D_1^{-1}$$

where

$$D_i = I_N - e_ie_i^T + l_{ii}e_ie_i^T, \quad P_{ic}^{-1} = I_N - \sum_{j=i+1}^{N} p_{ji}e_je_i^T, \quad P_{ir}^{-1} = I_N - \sum_{j=1}^{i-1} p_{ij}e_ie_j^T.$$

Using an additional $O(N^2)$ divisions one can also obtain the following two forms easily:

(3.3c) $$P_{G11}^{-1} = D^{-1}P_{N-1,c'}^{-1} \cdots P_{1c'}^{-1}, \quad P_{ic'}^{-1} = I_N - \sum_{j=i+1}^{N} p'_{ji}e_je_i^T, \; 1 \le i \le N-1,$$

(3.3d) $$P_{CR1}^{-1} = D^{-1}P_{Nr'}^{-1} \cdots P_{2r'}^{-1}, \quad P_{ir'}^{-1} = I_N - \sum_{j=1}^{i-1} p'_{ij}e_ie_j^T, \quad 2 \le i \le N$$

where

$$p'_{ji} = fl(p_{ji}/p_{ii}), \quad D = diag[p_{11}, \dots, p_{NN}].$$

Note the above are forward type of algorithms. In (3.3a) and (3.3b) P is gradually reduced to an identity matrix, whereas in (3.3c) and (3.3d) it is reduced first to a diagonal matrix. If we pre-multiply P first by D^{-1}, then the new matrix becomes a unit-diagonal lower triangular matrix and four new forms similar to (3.2a) through (3.2d) can be derived. By Theorem 3.2b we know that they are equivalent among themselves. We shall only give one of them in detail:

(3.3e) $$P_{G12}^{-1} = P_{N-1,c''}^{-1} \cdots P_{1c''}^{-1}D^{-1}, \; P_{ic''}^{-1} = I_N - \sum_{j=i+1}^{N} p''_{ji}e_je_i^T,$$
$$p''_{ji} = fl(p_{ji}/p_{jj}), \; 1 \le i \le N-1.$$

Finally P can also be reduced first to diagonal matrix by backward type of algorithms. We have

(3.3f) $$P_{Qc}^{-1} = D^{-1}Q_{1c} \cdots Q_{n-1,c}, \quad Q_{ic} = I_N - \sum_{j=i+1}^{N} q_{ji}e_je_i^T, \quad 1 \le i \le N-1,$$

(3.3g) $$P_{Qr}^{-1} = D^{-1}Q_{2r} \cdots Q_{Nr}\,, \quad Q_{ir} = I_N - \sum_{j=1}^{i-1} q_{ij}e_ie_j^T\,, \quad 2 \le i \le N$$

where

(3.3h) $$q_{i+k,i} = fl(\,(\,p_{i+k,i} - q_{i+k,i+k-1}p_{i+k-1,i} - \cdots - q_{i+k,i+1}p_{i+1,i}\,)/p_{ii}\,), \quad 1 \le k \le N-1\,, \quad 1 \le i \le N-k.$$

The following theorem can be proved using Lemma 2.3 and induction:

Theorem 3.3

(i) Given the matrix P whose error complexities satisfy Theorem 3.1, then the optimal order to find Q using (3.3h) is the left-to-right strategy for the summations. The computed Q has error complexities given as

$$\bar{q}_{i+k,i} = \bar{m}_{i+k,i}\,, \quad 1 \le k \le N-1\,, \quad 1 \le i \le N-k.$$

(ii) If the given right hand side vector b is such that $\bar{b}_i = \{1,0,0\}$, $1 \le i \le N$, then the computed

$$g = fl(\,P^{-1}b\,)$$

using any one of (3.3a) through (3.3g) is such that

$$\bar{g}_N = c_N\bar{\Delta}/c_{N*}\,, \quad \bar{g}_i = \bar{r}_{iN}\,, \quad 1 \le i \le N-1$$

provided the summations used to find g_i in (3.3b), (3.3d), (3.3g) are carried out , respectively, using the left-to-right strategy in each of the following expressions:

(3.3b1) $$g_i = fl(\,(\,b_i - p_{i1}g_1 - \cdots - p_{i,i-1}g_{i-1}\,)/p_{ii}\,)\,, \quad 1 \le i \le N,$$

(3.3d1) $$g_i = fl\,(h_i/p_{ii}\,)\,, \quad h_i = fl(\,b_i - p_{i1}h_1 - \cdots - p_{i,i-1}h_{i-1}\,)\,, \quad 1 \le i \le N\,,$$

(3.3g1) $$g_i = fl(\,(\,b_i - q_{i,i-1}b_{i-1} - \dots - q_{i1}b_1\,)/p_{ii}\,)\,, \quad 1 \le i \le N.$$

Methods for finding R^{-1} and U^{-1} are similar to those for finding L^{-1} and P^{-1}, respectively. We shall simply list them and give the resulting theorems without proofs.

Methods for R^{-1}:

(3.4a) $$R_{B1}^{-1} = R_{2c}^{-1} \dots R_{Nc}^{-1}\,, \quad R_{ic}^{-1} = I_N - \sum_{j=1}^{i-1} r_{ji}e_je_i^T\,, \quad 2 \le i \le N,$$

(3.4b) $$R_{B2}^{-1} = R_{1r}^{-1} \dots R_{N-1,r}^{-1}\,, \quad R_{ir}^{-1} = I_N - \sum_{j=i+1}^{N} r_{ij}e_ie_j^T\,, \quad 1 \le i \le N-1,$$

(3.4c) $$R_{F1}^{-1} = M'_{Nc} \cdots M'_{2c}\,, \quad M'_{ic} = I_N - \sum_{j=1}^{i-1} m'_{ji}e_je_i^T\,, \quad 2 \le i \le N,$$

(3.4d) $$R_{F2}^{-1} = M'_{N-1,r} \cdots M'_{1r}\,, \quad M'_{ir} = I_N - \sum_{j=i+1}^{N} m'_{ij}e_ie_j^T\,, \quad 1 \le i \le N-1$$

where M' is a unit-diagonal upper triangular matrix computed by the following formula:

(3.4e) $$m'_{i,i+k} = fl(\,r_{i,i+k} - m'_{i,i+1}r_{i+1,i+k} - \dots - m'_{i,i+k-1}r_{i+k-1,i+k}\,)\,,$$
$$1 \le k \le N-1\,, \quad 1 \le i \le N-k.$$

Theorem 3.4

(i) Given R whose error complexities satisfy Theorem 3.1, then the optimal order to evaluate (3.4e) is the left-to-right strategy. The computed M' is such that

$$\overline{m}'_{i,i+k} = \overline{m}_{i+k,i}\,, \quad 1 \le k \le N-1\,, \quad 1 \le i \le N-k.$$

(ii) given g whose error complexities satisfy Theorem 3.3, then the computed

$$x = fl(R^{-1}g)$$

using any one of (3.4a) through (3.4d) is such that

$$\bar{x}_i = \frac{c_i c_{i+1} \cdots c_N}{c_{i*} c_{i+1*} \cdots c_{N*}} \bar{\Delta}_{13}^{N-i} \bar{\Delta}, \quad 1 \le i \le N$$

provided the summations used to evaluate x_i in (3.4b) and (3.4d) are carried out in optimal order of the left-to-right strategy , respectively, using the following expressions:

(3.4b1) $$x_N = g_N, \quad x_i = fl(g_i - r_{iN} x_N - \ \cdots \ - r_{i,i+1} x_{i+1}), \quad 1 \le i \le N-1,$$

(3.4d1) $$x_i = fl(g_i - m'_{i,i+1} g_{i+1} - \cdots - m'_{iN} g_N), \quad 1 \le i \le N.$$

<u>Methods for U^{-1}:</u>

(3.5a) $$U_{B1}^{-1} = D_{U1}^{-1} U_{2c} D_{U2}^{-1} \cdots U_{Nc}^{-1} D_{UN}^{-1},$$

(3.5b) $$U_{B2}^{-1} = D_{U1}^{-1} U_{1r}^{-1} \cdots D_{U,N-1}^{-1} U_{N-1,r}^{-1} D_{UN}^{-1}$$

where

$$D_{Ui} = I_N - e_i e_i^T + u_{ii} e_i e_i^T, \quad U_{ic}^{-1} = I_N - \sum_{j=1}^{i-1} u_{ji} e_j e_i^T, \quad U_{ir}^{-1} = I_N - \sum_{j=i+1}^{N} u_{ij} e_i e_j^T.$$

(3.5c) $$U_{B3}^{-1} = D_U^{-1} U_{2c'}^{-1} \cdots U_{Nc'}^{-1}, \ U_{ic'}^{-1} = I_N - \sum_{j=1}^{i-1} u'_{ji} e_j e_i^T, \quad 2 \le i \le N,$$

(3.5d) $$U_{B4}^{-1} = D_U^{-1} U_{1r'}^{-1} \cdots U_{N-1,r'}^{-1}, \ U_{ir'}^{-1} = I_N - \sum_{j=i+1}^{N} u'_{ij} e_i e_j^T, \quad 1 \le i \le N-1,$$

where

$$u'_{ji} = fl(u_{ij}/u_{jj}), \quad D_U = diag[u_{ii}, \dots, u_{NN}].$$

(3.5e) $$U_{B5}^{-1} = U_{2c''}^{-1} \dots U_{Nc''}^{-1} D_U^{-1}, \quad U_{ic''}^{-1} = I_N - \sum_{j=1}^{i-1} u''_{ji} e_j e_i^T, \; u''_{ji} = fl(u_{ji}/u_{jj}), \quad 2 \le i \le N.$$

(3.5f) $$U_{F1}^{-1} = D_U^{-1} Q'_{Nc} \cdots Q'_{2c}, \quad Q'_{ic} = I_N - \sum_{j=1}^{i-1} q'_{ji} e_j e_i^T, \quad 2 \le i \le N,$$

(3.5g) $$U_{F2}^{-1} = D_U^{-1} Q'_{N-1,r} \cdots Q'_{1r}, \quad Q'_{ir} = I_N - \sum_{j=i+1}^{N} q'_{ij} e_i e_j^T, \quad 1 \le i \le N-1$$

where Q' is a unit-diagonal upper triangular matrix computed by the following formula:

(3.5h) $$q'_{i,i+k} = fl((u_{i,i+k} - q'_{i,i+1} u_{i+1,i+k} - \dots - q'_{i,i+k-1} u_{i+k-1,i+k})/u_{i+k,i+k}).$$

Theorem 3.5

(i) Given the matrix U whose error complexities satisfy Theorem 3.1, then the optimal order to find Q' using (3.5h) is the left-to-right strategy for the summations. The computed Q' has error complexities given as

$$\bar{q}'_{i,i+k} = c_{i*}\left(\prod_{j=i}^{i+k-1} c_j\right) \bar{\Delta}_{13}^{k-1} \bar{\Delta} / c_{i+k*}, \quad 1 \le k \le N-1, \quad 1 \le i \le N-k.$$

(ii) Given f whose error complexities satisfy Theorem 3.2b, then the computed

$$x' = fl(U^{-1} f)$$

using any one of (3.5a) through (3.5g) is such that

$$\bar{x}'_i = \bar{x}_i, \quad 1 \le i \le N$$

provided the summations used to evaluate x'_i in (3.5b), (3.5d), (3.5g) are carried out, respectively, using the left-to-right strategy in each of the following expressions:

(3.5b1) $$x'_i = fl((f_i - u_{iN}x'_N - \dots - u_{i,i+1}x'_{i+1})/u_{ii}),\ 1 \le i \le N,$$

(3.5d1) $$x'_i = fl(y_i/u_{ii}),\ y_i = fl(f_i - u'_{iN}y_N - \dots - u'_{i,i+1}y_{i+1}),\ \ 1 \le i \le N,$$

(3.5g1) $$x'_i = fl((f_i - q'_{i,i+1}f_{i+1} - \dots - q'_{iN}f_N)/u_{ii}),\ 1 \le i \le N.$$

By Theorem 3.5 we conclude that the class of inverse decomposition methods based on finding the triangular factors of the matrix A followed by explicitly or implicitly inverting the triangular factors are equivalent among themselves in terms of our error complexity measures.

References

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

[2] 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.

Appendix I. Proof of Theorem 3.1

We prove by induction on k and i. For $k = i = 1$ we have

$$u_{1j} = a_{ij}\,,\quad 1 \le j \le N;\quad l_{t1} = fl\,(a_{t1}/u_{11}) = a_{t1}\Delta/a_{11}\,,\quad 2 \le t \le N.$$

Hence

$$\bar{u}_{11} = c_{1*}\,, \bar{u}_{1j} = c_1\,, 2 \le j \le N,$$

$$\bar{l}_{t1} = \bar{a}_{t1}\bar{\Delta}/\bar{a}_{11} = c_1\bar{\Delta}/c_{1*}\,,\quad 2 \le t \le N$$

and the theorem is true. Assume the theorem is true for $k = i = r - 1$. For $k = i = r$ we assume that in the computation of u_{kj} the given l_{kt} and u_{tj} for $1 \le t \le k - 1$ are calculated by the optimal order and so u_{kj} can be considered as the computed result of

$$z = y_1 + y_2 + \ldots + y_k\,,\quad y_1 = a_{kj}\,,\quad y_{t+1} = fl\,(l_{kt}u_{tj})\,,\quad 1 \le t \le k - 1.$$

Now by assumption

$$\bar{y}_1 = c_1\,,\quad \bar{l}_{kt} = c_t\bar{\Delta}/c_{t*}\,,\quad \bar{u}_{tj} = \frac{c_t}{c_{1*}c_{2*}\ldots c_{t-1*}}\,.$$

Hence

$$\bar{y}_1 = c_1\,,\quad \bar{y}_{t+1} = \frac{c_t c_t \bar{\Delta}^2}{c_{1*}c_{2*}\ldots c_{t*}}\,,\quad 1 \le t \le k - 1.$$

Now if exact additions were possible, then

$$z = \frac{1}{z_d}\sum_{t=1}^{k} z_{ni}$$

where

$$\bar{z}_d = c_{1*}c_{2*}\ldots c_{k-1*},\quad \bar{z}_{n1} = c_1 c_{1*}c_{2*}\ldots c_{k-1*}\,, \bar{z}_{n,t+1} = c_t c_t c_{t+1*}c_{t+2*}\ldots c_{k-1*}\bar{\Delta}^2\,,\quad 1 \le t \le k - 1.$$

By definition we have

$$c_{t+1*} = c_{t+1} = c_t c_t \overline{\Delta}_{13} = c_t c_t \{2,3,4\}.$$

Hence

$$\lambda(c_{t+1*}) = \lambda(c_{t+1}) = 2\lambda^2(c_t)\,, \quad \sigma(c_{t+1*}) = \sigma(c_{t+1}) = 2\sigma(c_t) + 3 > \sigma(c_t) + 1.$$

Applying the above equations to $\overline{z}_{n,t+1}$ and $\overline{z}_{nt}$ we have

$$\frac{\lambda(z_{n2})}{\lambda(z_{n1})} = 1\,, \quad \frac{\lambda(z_{n,t+1})}{\lambda(z_{n,t})} = \frac{\lambda(c_t)}{\lambda^2(c_{t-1})} = 2\,,$$
$$\sigma(z_{n,i+1}) - \sigma(z_{ni}) = \sigma(c_i) - 2\sigma(c_{i-1}) = 3 > 1.$$

The above equations satisfy the conditions of Lemma 2.3, hence by the lemma the best strategy in evaluating

$$u_{kj} = fl\,(y_1 + y_2 + \dots + y_k)$$

is by the left-to-right strategy. Thus by Lemma 2.3 and the induction assumption we have

$$\overline{u}_{kj} = (\overline{y}_1\overline{\Delta}^{k-1} + \overline{y}_2\overline{\Delta}^{k-1} + \dots + \overline{y}_{k-1}\overline{\Delta}^2) + \overline{y}_k\overline{\Delta}$$
$$= \frac{\overline{c}_{k-1}\overline{\Delta}}{c_{1*}c_{2*}\dots c_{k-2*}} + \frac{c_{k-1}c_{k-1}\overline{\Delta}^3}{c_{1*}c_{2*}\dots c_{k-1*}} = \frac{c_k}{c_{1*}c_{2*}\dots c_{k-1*}}\,.$$

Similar reasoning can be used to show the remaining part of the theorem for l_{ji}'s. This completes our proof. Q.E.D.

Appendix I. Proof of Theorem 3.1

We prove by induction on k and i. For $k = i = 1$ we have

$$u_{1j} = a_{ij}\,, \quad 1 \le j \le N; \quad l_{t1} = fl\,(a_{t1}/u_{11}) = a_{t1}\Delta/a_{11}\,, \quad 2 \le t \le N.$$

Hence

$$\bar{u}_{11} = c_{1*}\,, \bar{u}_{1j} = c_1\,, 2 \le j \le N,$$

$$\bar{l}_{t1} = \bar{a}_{t1}\bar{\Delta}/\bar{a}_{11} = c_1\bar{\Delta}/c_{1*}\,, \quad 2 \le t \le N$$

and the theorem is true. Assume the theorem is true for $k = i = r - 1$. For $k = i = r$ we assume that in the computation of u_{kj} the given l_{kt} and u_{tj} for $1 \le t \le k - 1$ are calculated by the optimal order and so u_{kj} can be considered as the computed result of

$$z = y_1 + y_2 + \ \dots\ + y_k\,, \quad y_1 = a_{kj}\,, \quad y_{t+1} = fl\,(l_{kt}u_{tj})\,, \quad 1 \le t \le k - 1.$$

Now by assumption

$$\bar{y}_1 = c_1\,, \quad \bar{l}_{kt} = c_t\bar{\Delta}/c_{t*}\,, \quad \bar{u}_{tj} = \frac{c_t}{c_{1*}c_{2*}\ \dots\ c_{t-1*}}\,.$$

Hence

$$\bar{y}_1 = c_1\,, \quad \bar{y}_{t+1} = \frac{c_t c_t \bar{\Delta}^2}{c_{1*}c_{2*}\ \dots\ c_{t*}}\,, \quad 1 \le t \le k - 1.$$

Now if exact additions were possible, then

$$z = \frac{1}{z_d}\sum_{t=1}^{k} z_{ni}$$

where

$$\bar{z}_d = c_{1*}c_{2*}\ \dots\ c_{k-1*}, \quad \bar{z}_{n1} = c_1c_{1*}c_{2*}\ \dots\ c_{k-1*}\,, \bar{z}_{n,t+1} = c_tc_tc_{t+1*}c_{t+2*}\ \dots\ c_{k-1*}\bar{\Delta}^2\,, \quad 1 \le t \le k - 1.$$

By definition we have

$$c_{t+1*} = c_{t+1} = c_t c_t \overline{\Delta}_{13} = c_t c_t \{2,3,4\}.$$

Hence

$$\lambda(c_{t+1*}) = \lambda(c_{t+1}) = 2\lambda^2(c_t)\,, \quad \sigma(c_{t+1*}) = \sigma(c_{t+1}) = 2\sigma(c_t) + 3 > \sigma(c_t) + 1.$$

Applying the above equations to $\bar{z}_{n,t+1}$ and $\bar{z}_{nt}$ we have

$$\frac{\lambda(z_{n2})}{\lambda(z_{n1})} = 1\,, \quad \frac{\lambda(z_{n,t+1})}{\lambda(z_{nt})} = \frac{\lambda(c_t)}{\lambda^2(c_{t-1})} = 2\,,$$
$$\sigma(z_{n,i+1}) - \sigma(z_{ni}) = \sigma(c_i) - 2\sigma(c_{i-1}) = 3 > 1.$$

The above equations satisfy the conditions of Lemma 2.3, hence by the lemma the best strategy in evaluating

$$u_{kj} = fl\,(y_1 + y_2 + \dots + y_k)$$

is by the left-to-right strategy. Thus by Lemma 2.3 and the induction assumption we have

$$\bar{u}_{kj} = (\bar{y}_1 \overline{\Delta}^{k-1} + \bar{y}_2 \overline{\Delta}^{k-1} + \dots + \bar{y}_{k-1} \overline{\Delta}^2) + \bar{y}_k \overline{\Delta}$$
$$= \frac{\bar{c}_{k-1} \overline{\Delta}}{c_{1*} c_{2*} \dots c_{k-2*}} + \frac{c_{k-1} c_{k-1} \overline{\Delta}^3}{c_{1*} c_{2*} \dots c_{k-1*}} = \frac{c_k}{c_{1*} c_{2*} \dots c_{k-1*}}\,.$$

Similar reasoning can be used to show the remaining part of the theorem for l_i's. This completes our proof. Q.E.D.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA TM-102036 ICOMP-89-11	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle On the Equivalence of a Class of Inverse Decomposition Algorithms for Solving Systems of Linear Equations		5. Report Date May 1989
		6. Performing Organization Code
7. Author(s) Nai-kuan Tsao		8. Performing Organization Report No. E-4785
		10. Work Unit No. 505-62-21
9. Performing Organization Name and Address National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135-3191		11. Contract or Grant No.
		13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546-0001		14. Sponsoring Agency Code

15. Supplementary Notes

Nai-kuan Tsao, Wayne State University, Detroit, Michigan 48202 and Institute for Computational Mechanics in Propulsion, Lewis Research Center (work funded under Space Act Agreement C99066G).
Space Act Monitor: Louis A. Povinelli.

16. Abstract

A class of direct inverse decomposition algorithms for solving systems of linear equations is presented. Their behavior in the presence of round-off errors is analyzed. It is shown that under some mild restrictions on their implementation, the class of direct inverse decomposition algorithms presented are equivalent in terms of our error complexity measures.

17. Key Words (Suggested by Author(s)) Gaussian elimination, Gauss-Jordan method; Triangular decomposition; Error complexity; Doolittle method; Crout method	18. Distribution Statement Unclassified – Unlimited Subject Category 64

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

NASA FORM 1626 OCT 86

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