Electronic Transactions on Numerical Analysis.
Volume 4, pp. 125-137, September 1996.
Copyright © 1996, Kent State University.
ISSN 1068-9613.

ON THE SOLUTION OF CAUCHY SYSTEMS OF EQUATIONS [*]

D. CALVETTI[†] AND L. REICHEL[‡]

Dedicated to Gerhard Opfer on the occasion of his 60th birthday.

Abstract. The development of fast solution methods for linear systems of equations with a Cauchy matrix has recently received considerable attention. This note presents new solution methods based on a modification of an inversion formula described by Gastinel [8].

Key words. fast algorithm, linear system, Cauchy matrix, Hankel matrix, Toeplitz matrix.

AMS subject classification. 65F10.

1. Introduction.

The present note describes numerical methods for the fast and accurate solution of linear systems of equations

$$C_n \boldsymbol{x} = \boldsymbol{b}, \qquad C_n \in \mathbb{C}^{n\times n}, \qquad \boldsymbol{x}, \boldsymbol{b} \in \mathbb{C}^n, \tag{1.1}$$

where $C_n = [c_{ij}]_{i,j=1}^n$ is a Cauchy matrix, i.e., its entries are of the form

$$c_{ij} = \frac{1}{s_i - t_j}, \qquad s_i, t_j \in \mathbb{C}, \qquad 1 \le i, j \le n. \tag{1.2}$$

We assume throughout this paper that the $2n$ nodes s_i and t_j are pairwise distinct. Then it follows from the well-known determinant formula

$$\det C_n = \frac{\prod_{1\le j<i\le n}(s_i - s_j) \cdot \prod_{1\le j<i\le n}(t_i - t_j)}{\prod_{i,j=1}^n (s_i - t_j)}, \tag{1.3}$$

see, e.g., [6, pp. 268-269], that the matrix C_n is non-singular.

EXAMPLE 1.1. Let $a > 0$ and b be real scalars such that b/a is not a negative integer smaller than -1. Define the nodes $s_i = ia + b$ and $t_j = -ja$. Then C_n is a real symmetric Hankel matrix. In particular, for $a = 1$ and $b = -1$, C_n is a Hilbert matrix.□

EXAMPLE 1.2. Let $a \neq 0$ and b be scalars such that b/a is not an integer. Define the nodes $s_i = ai + b$ and $t_j = aj$. Then C_n is a Toeplitz matrix. □

A matrix $\hat{C}_n = [\hat{c}_{ij}]_{i,j=1}^n$ is said to be of Cauchy-type if its entries are of the form

$$\hat{c}_{ij} = \frac{u_i v_j}{s_i - t_j}, \tag{1.4}$$

where the nodes satisfy the same conditions as in (1.2), and $u_i, v_j \in \mathbb{C}$ are nonvanishing scalars. Let $U = \mathrm{diag}(u_1, u_2, \ldots, u_n)$ and $V = \mathrm{diag}(v_1, v_2, \ldots, v_n)$. Then $\hat{C}_n$ can be factored according to

$$\hat{C}_n = U C_n V, \tag{1.5}$$

[*] Received February 27, 1996. Accepted for publications August 8, 1996. Communicated by A. Ruttan.

[†] Stevens Institute of Technology, Department of Mathematical Sciences, Hoboken, NJ 07030. Research supported in part by NSF grant DMS-9404692. (na.calvetti@na-net.ornl.gov).

[‡] Kent State University, Department of Mathematics and Computer Science, Kent, OH 44242. Research supported in part by NSF grant DMS-9404706. (reichel@mcs.kent.edu).

where C_n is defined by (1.2). The solution methods described in this note can trivially also be applied to the solution of linear systems of equations with a matrix of Cauchy-type by using the factorization (1.5).

Linear systems of equations with matrices of Cauchy-type arise from the discretization of singular integral equations; see, e.g., [9, 20, 22]. Moreover, linear systems of equations with a Vandermonde or a Chebyshev-Vandermonde matrix can be transformed into a linear system with a matrix of Cauchy-type by using the discrete Fourier transform. Accurate solution methods for linear systems with a Cauchy-type matrix therefore can be used to compute accurate solutions of linear systems with a Vandermonde or Chebyshev-Vandermonde matrix; see [4, 11, 14].

We remark that linear systems of equations of the form (1.1) also arise from interpolation at the nodes $\{s_i\}_{i=1}^n$ by rational functions with prescribed simple poles $\{t_j\}_{j=1}^n$ when using the basis $\{1/(z-t_j)\}_{j=1}^n$. However, this basis of rational functions can be quite ill-conditioned, and the use of the basis $\{(z-t_1)^{-1}, (z-t_1)^{-1}(z-t_2)^{-1}, \dots, \prod_{j=1}^n (z-t_j)^{-1}\}$ is preferable since it is often better conditioned; see [19] for a discussion.

The solution of the linear systems (1.1) by Gaussian elimination with partial pivoting requires $O(n^3)$ arithmetic operations. By using the structure of the matrix C_n, faster solution methods can be devised. For instance, Gastinel [8] described how the entries of the inverse C_n^{-1} of the matrix C_n can be computed in only $O(n^2)$ arithmetic operations. This result can also be found in [6, p. 288].

Gerasoulis et al. [9, 10] presented an algorithm that allows the multiplication of an $n \times n$ matrix of Cauchy-type with a vector using only $O(n \log^2 n)$ arithmetic operations. Related algorithms have also been considered by Gohberg and Olshevsky [12]. In view of the fact that the inverse of a Cauchy-type matrix is also a matrix of Cauchy-type, these algorithms can be used to solve linear systems of equations with a Cauchy-type matrix in only $O(n \log^2 n)$ arithmetic operations; see Gohberg and Olshevsky [12], as well as Bini and Pan [2], for details. Unfortunately, these algorithms are numerically unstable for many distributions of nodes s_i and t_j.

The multipole method described by Greengard and Rokhlin [13] can be used to solve Cauchy systems (1.1) in only $O(n)$ arithmetic operations for a prescribed accuracy. The operation count increases with the desired accuracy and depends on the distribution of the nodes that define the Cauchy matrix. Multipole methods are competitive for very large values of n only.

Due to the possible instability of the methods that require $O(n \log^2 n)$ arithmetic operations, and the uncompetitiveness of multipole methods for moderate values of n, the development of algorithms that require $O(n^2)$ arithmetic operations continues to receive considerable attention. Recently, Boros et al. [5] presented several algorithms for the triangular factorization of a Cauchy matrix, or the inverse of such a matrix, in $O(n^2)$ arithmetic operations. The numerical examples in [5] show that these algorithms give high accuracy for linear systems (1.1) with certain Cauchy matrices and right-hand sides, but they yield poor accuracy for other Cauchy systems. This is also demonstrated by the computed examples in Section 4. Given a particular linear system of equations with a Cauchy matrix, it is not clear which one of these algorithms determines the most accurate approximate solution.

The purpose of this note is to present a modification of the inversion formula described by Gastinel [8] and to discuss solution methods based on this formula. The solution methods so obtained require $O(n^2)$ arithmetic operations and they are well suited for parallel computation. Section 2 presents the new inversion formula. A round-off error analysis for this inversion formula is carried out in Section 3, and solution methods for Cauchy systems based on this inversion formula are discussed. These solution methods differ in whether iterative refinement is used. Computed examples are presented in Section 4, and concluding remarks can be found in Section 5.

2. A new inversion formula.

Our solution method is based on the following inversion formula.

THEOREM 2.1. *The solution* $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T$ *of the linear system (1.1) with right-hand side vector* $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)^T$ *and matrix elements (1.2) is given by*

$$x_j = \frac{f(t_j)}{g'(t_j)} \left\{ -\gamma + \sum_{i=1}^{n} \left(\frac{g(s_i)}{f'(s_i)} \cdot \frac{b_i - \gamma}{t_j - s_i} \right) \right\}, \qquad 1 \leq j \leq n, \tag{2.1}$$

where $\gamma \in \mathbb{C}$ *is an arbitrary constant, and* f *and* g *are the polynomials*

$$f(s) = \prod_{i=1}^{n} (s - s_i), \qquad g(t) = \prod_{j=1}^{n} (t - t_j). \tag{2.2}$$

Proof. Let h be a polynomial of degree at most $n-1$ and let $\gamma \in \mathbb{C}$. Introduce the partial fraction expansion

$$\frac{h(z) - \gamma f(z)}{g(z)} = -\gamma + \sum_{j=1}^{n} \frac{x_j}{z - t_j}, \tag{2.3}$$

$$x_j = \frac{h(t_j) - \gamma f(t_j)}{g'(t_j)}, \tag{2.4}$$

where g' denotes the derivative of g. In view of $f(s_i) = 0$, we obtain from (2.3) that

$$\frac{h(s_i)}{g(s_i)} = -\gamma + \sum_{j=1}^{n} \frac{x_j}{s_i - t_j}, \qquad 1 \leq i \leq n. \tag{2.5}$$

The polynomial h is uniquely determined by the interpolation conditions

$$h(s_i) = g(s_i)(b_i - \gamma), \qquad 1 \leq i \leq n. \tag{2.6}$$

Substitution of (2.6) into (2.5) shows that the vector $\boldsymbol{x}$ with components given by (2.4) solves (1.1). Express the polynomial h determined by (2.6) in Lagrange form

$$h(z) = \sum_{i=1}^{n} g(s_i)(b_i - \gamma)\ell_i(z), \qquad \ell_i(z) = \frac{f(z)}{f'(s_i)} \cdot \frac{1}{z - s_i}, \tag{2.7}$$

and substitute (2.7) into (2.4). This establishes the theorem. □

REMARK 2.1. Setting $\gamma = 0$ in (2.1) shows that the elements of $C_n^{-1} = [\tilde{c}_{ij}]_{i,j=1}^{n}$ are given by

$$\tilde{c}_{ij} = \frac{f(t_i)}{g'(t_i)} \cdot \frac{1}{t_i - s_j} \cdot \frac{g(s_j)}{f'(s_j)}. \tag{2.8}$$

Formula (2.8) was shown by Gastinel [8] and demonstrates that the inverse matrix C_n^{-1} is of Cauchy-type. □

The parameter γ can be selected so that formula (2.1) yields a very accurate solution for certain right-hand side vectors $\boldsymbol{b}$. Introduce the vector $\boldsymbol{e} = (1, 1, \ldots, 1)^T$.

Example 2.1. Let $\boldsymbol{b} = \alpha \boldsymbol{e}$ for some constant α. Then letting $\gamma = \alpha$ in (2.1) yields $x_i = -\alpha f(t_i)/g'(t_i)$, $1 \le i \le n$. Thus, the solution is obtained by multiplying $\boldsymbol{b}$ by a diagonal matrix. By Lemma 3.1 below the quotients $f(t_i)/g'(t_i)$, $1 \le i \le n$, can be evaluated with high relative accuracy, and therefore so can the entries of the computed solution. □

Example 2.2. Assume that

$$t_n < t_{n-1} < \ldots < t_1 < s_1 < s_2 \ldots < s_n. \tag{2.9}$$

Then it follows from (2.8) that

$$\mathrm{sign}(\tilde{c}_{ij}) = (-1)^{i+j}, \qquad 1 \le i, j \le n.$$

In particular, the Hilbert matrix considered in Example 1.1 satisfies (2.9). Let the entries of the vector $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)^T$ satisfy

$$\mathrm{sign}((-1)^k w_{2i-1}) \ge 0, \qquad \mathrm{sign}((-1)^k w_{2i}) \le 0,$$

for all i and some integer k. Then the matrix-vector product $C_n^{-1}\boldsymbol{w}$ can be evaluated with high relative accuracy, since no cancellation of significant digits occurs.

The inequalities (2.9) imply that all factors in the determinant formula (1.3) are positive, and it follows that C_n is totally positive; see, e.g., Minc [17] for properties of totally positive matrices. □

Example 2.3. Assume that the nodes satisfy (2.9). Let $\boldsymbol{b} = (2, 1, 2, 1, \ldots\ )^T$ and assume that $1 \le \gamma \le 2$. Then the vector $\boldsymbol{w} = \boldsymbol{b} - \gamma\boldsymbol{e}$ satisfies the conditions of Example 2.2. The solution $\boldsymbol{x}$ computed by formula (2.1) is the sum of two vectors. By Lemma 3.1 below the quotients $f(t_i)/g'(t_i)$ and the elements $\tilde{c}_{ij}$ of C_n^{-1} can be evaluated with high relative accuracy. Therefore the vectors $\boldsymbol{x}^{(1)} = -\gamma(f(t_1)/g'(t_1), f(t_2)/g'(t_2), \ldots, f(t_n)/g'(t_n))^T$ and $\boldsymbol{x}^{(2)} = C_n^{-1}\boldsymbol{w}$, whose sum yields the solution, can be evaluated with high relative accuracy. Moreover, if n is odd, then the corresponding components of the vectors $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ are of the same sign and their sum can be evaluated with high relative accuracy. □

Example 2.3 suggests that the parameter γ in (2.1) be chosen so that the vector $\boldsymbol{b} - \gamma\boldsymbol{e}$ contains entries of different sign. In the computed examples we therefore choose γ to be the median of the entries of $\boldsymbol{b}$. This choice guarantees that the number of nonpositive and nonnegative entries of the vector $\boldsymbol{b} - \gamma\boldsymbol{e}$ differ by at most one. We remark that another possibility is to let γ be the average of the entries of $\boldsymbol{b}$. The latter choice of γ splits the vector $\boldsymbol{b}$ into the orthogonal vectors $\gamma\boldsymbol{e}$ and $\boldsymbol{b} - \gamma\boldsymbol{e}$. We found that the former choice often gives slightly higher accuracy than the latter. The median is computed by first ordering the entries of $\boldsymbol{b}$ monotonically. This can be done in $O(n \log n)$ time steps with one processor and in $O(\lceil \log^2 n \rceil)$ time steps with n processors; see Batcher [1] for details on the latter. Here $\lceil s \rceil$ denotes the smallest integer larger than or equal to $s \in \mathbb{R}$.

Given γ, the evaluation of all the components x_i of the solution using (2.1) can be carried out in $O(n^2)$ arithmetic operations. The precise operation count depends on the representation used for the polynomial h. We choose to represent the polynomial h given by (2.7) in Lagrange form because this yields high accuracy. In order to avoid overflow and underflow during the computations, we evaluate the quotients $f(t_i)/g'(t_i)$ by forming products of the quotients $(t_i - s_j)/(t_i - t_j)$ for $1 \le j < i$ and $i < j \le n$. The quotients $g(s_j)/f'(s_j)$ are evaluated analogously. When n processors are available, these computations can be carried out in $O(n)$ time steps.

3. Round-off error analysis. The solution computed by formula (2.1) is contaminated by propagated round-off errors. This section discusses the effect of the round-off errors on the accuracy of the computed solution, and we consider the benefits of iterative improvement. All computations were carried out with single precision arithmetic except when explicitly stated otherwise, and we assume that the standard model for floating point operations holds, i.e.,

$$fl(a \circ b) = (a \circ b)(1+\delta), \qquad |\delta| \leq u, \qquad \circ \in \{+,-,*,/\}, \tag{3.1}$$

for all scalars a and b. Throughout this paper u denotes the unit round-off.

Pairwise summation, also known as Babuska summation, is used to evaluate the sums that occur in matrix-vector products and in the evaluation of x_j according to (2.1), except when explicitly stated otherwise in Example 4.4. Pairwise summation evaluates a sum $\sum_{j=1}^{n} \alpha_j$ by first adding the pairs

$$\hat{\alpha}_j = \alpha_{2j-1} + \alpha_{2j}, \qquad 1 \leq j \leq \lfloor n/2 \rfloor,$$

where $\lfloor s \rfloor$ denotes the largest integer smaller than or equal to $s \in \mathbb{R}$. When n is odd, we define $\hat{\alpha}_{(n+1)/2} = \alpha_n$. Pairwise summation is then repeated recursively starting with the $\hat{\alpha}_j$, $1 \leq j \leq \lfloor (n+1)/2 \rfloor$. The sum is obtained in $\lceil \log_2 n \rceil$ time steps. Pairwise and other summation methods have recently been discussed by Higham [15].

Let $\boldsymbol{x} = (x_1, x_2, \dots, x_n)^T$ denote the solution of (1.1), and let a numerical method produce an approximate solution $\hat{\boldsymbol{x}}$. Introduce the associated residual vector $\hat{\boldsymbol{r}} = \boldsymbol{b} - C_n\hat{\boldsymbol{x}}$. Let $\|\cdot\|$ denote the maximum norm on $\mathbb{C}^n$, as well as the associated induced matrix norm on $\mathbb{C}^{n\times n}$. We define the condition number $\kappa(C_n) = \|C_n\|\,\|C_n^{-1}\|$. Following Jankowski and Wozniakowski [16], we say that the numerical method is *stable* if

$$\|\hat{\boldsymbol{x}} - \boldsymbol{x}\| \leq u d_1(n) \kappa(C_n) \|\boldsymbol{x}\|, \tag{3.2}$$

where $d_1(n)$ is a function of n only. In particular, $d_1(n)$ is independent of C_n and $\boldsymbol{b}$. The numerical method is said to be *well behaved* if

$$\|\hat{\boldsymbol{r}}\| \leq u d_2(n) \|C_n\| \|\boldsymbol{x}\|, \tag{3.3}$$

where $d_2(n)$ is a function of n only. It is easy to see that a well-behaved method is stable, but the converse is in general not true.

Lemma 3.1. *Let $\hat{\eta}_j$ denote the computed value of $f(t_j)/g'(t_j)$, $\hat{\xi}_i$ the computed value of $g(s_i)/f'(s_i)$, and $\hat{\gamma}_{ij}$ the computed value of the entry $\tilde{c}_{ij}$ of C_n^{-1}. Assume that $4nu < 1$. Then*

$$\hat{\eta}_j = \frac{f(t_j)}{g'(t_j)}(1+\theta_{4n-3}^{(j)}u), \quad \hat{\xi}_i = \frac{g(s_i)}{f'(s_i)}(1+\hat{\theta}_{4n-3}^{(i)}u), \quad \hat{\gamma}_{ij} = \tilde{c}_{ij}(1+\theta_{4n}^{(i,j)}), \tag{3.4}$$

where $|\theta_k^{(j)}|$, $|\hat{\theta}_k^{(i)}|$ and $|\theta_k^{(i,j)}|$ are bounded by $ku/(1-ku)$ for all i and j.

Proof. Assume that $|\epsilon_k| < u$ for $1 \leq k \leq m$, and that $mu < 1$. Then

$$\prod_{k=1}^{m}(1+\epsilon_k) = 1 + \theta_m, \tag{3.5}$$

where $|\theta_m| \leq mu/(1-mu)$. The equalities (3.4) now follow from application of (3.1) and (3.5) to (2.2) and (2.8). □

The lemma establishes that the quotients $f(t_j)/g'(t_j)$ and $g(s_i)/f'(s_i)$, as well as the entries $\tilde{c}_{ij}$, can be evaluated with high relative accuracy. Therefore, we will henceforth assume that these quantities are evaluated exactly. The right-hand side vector $\boldsymbol{b}$ is also assumed to be exact. We will study the propagation of round-off errors when formula (2.1) is used for different right-hand side vectors $\boldsymbol{b}$ and different choices of the parameter γ. This sheds light on the numerical properties of formula (2.1) for general right-hand side vectors. We denote the computed approximate solution obtained from (2.1) by $\hat{\boldsymbol{x}}^{(0)}$, and define the associated error $\boldsymbol{e}^{(0)} = \hat{\boldsymbol{x}}^{(0)} - \boldsymbol{x}$ and residual error $\boldsymbol{r}^{(0)} = \boldsymbol{b} - C_n\hat{\boldsymbol{x}}^{(0)}$.

THEOREM 3.2. *Let $\boldsymbol{b} = \alpha\boldsymbol{e}$ and $\gamma = \alpha$ in (2.1). Then the error $\boldsymbol{e}^{(0)}$ in the approximate solution $\hat{\boldsymbol{x}}^{(0)}$ computed by (2.1) and the residual vector $\boldsymbol{r}^{(0)}$ satisfy*

$$\|\boldsymbol{e}^{(0)}\| \leq u\|\boldsymbol{x}\|, \tag{3.6}$$

$$\|\boldsymbol{r}^{(0)}\| \leq u\|C_n\|\|\boldsymbol{x}\|. \tag{3.7}$$

Let instead $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)^T$ be such that $|\sum_{j=1}^n \tilde{c}_{ij}b_j| = \sum_{j=1}^n |\tilde{c}_{ij}b_j|$, where $\tilde{c}_{ij}$ are the elements of C_n^{-1} and let $\gamma = 0$. Then

$$\|\boldsymbol{e}^{(0)}\| \leq u(1 + \lceil\log_2 n\rceil + O(u))\|\boldsymbol{x}\|, \tag{3.8}$$

$$\|\boldsymbol{r}^{(0)}\| \leq u(1 + \lceil\log_2 n\rceil + O(u))\|C_n\|\|\boldsymbol{x}\|. \tag{3.9}$$

Finally, let $\boldsymbol{b} \in \mathbb{C}^n$ be a general vector, and let $\gamma = 0$ in (2.1). Then

$$\|\boldsymbol{e}^{(0)}\| \leq u(1 + \lceil\log_2 n\rceil + O(u))\kappa(C_n)\|\boldsymbol{x}\|, \tag{3.10}$$

$$\|\boldsymbol{r}^{(0)}\| \leq u(1 + \lceil\log_2 n\rceil + O(u))\kappa(C_n)\|\boldsymbol{b}\|. \tag{3.11}$$

Proof. We first show (3.6). We have $b_i = \alpha$. Introduce $\eta_j = f(t_j)/g'(t_j)$. The components of the exact solution are given by $x_i = -\alpha\eta_i$, and the components of the computed solution are $\hat{x}_i^{(0)} = -fl(\alpha\eta_i)$. By (3.1), we have $|\hat{x}_i^{(0)} - x_i| = |fl(\alpha\eta_i) - \alpha\eta_i| \leq u|\alpha\eta_i| = u|x_i|$, and (3.6) follows.

We turn to the proof of (3.10). From $\hat{x}_i^{(0)} = fl(\sum_{j=1}^n fl(\tilde{c}_{ij}b_j))$, we obtain

$$|\hat{x}_i^{(0)} - x_i| \leq |fl(\sum_{j=1}^n fl(\tilde{c}_{ij}b_j)) - \sum_{j=1}^n fl(\tilde{c}_{ij}b_j)| + |\sum_{j=1}^n fl(\tilde{c}_{ij}b_j) - \sum_{j=1}^n \tilde{c}_{ij}b_j|.$$

For pairwise summation we have the bound

$$|fl(\sum_{j=1}^n fl(\tilde{c}_{ij}b_j)) - \sum_{j=1}^n fl(\tilde{c}_{ij}b_j)| \leq (1+u)\delta_n \sum_{j=1}^n |\tilde{c}_{ij}b_j|,$$

where

$$\delta_n = u\lceil\log_2 n\rceil/(1 - u\lceil\log_2 n\rceil); \tag{3.12}$$

see [15]. We obtain

$$|\hat{x}_i^{(0)} - x_i| \leq (u + (1+u)\delta_n) \sum_{j=1}^n |\tilde{c}_{ij}b_j|, \tag{3.13}$$

and, therefore,

$$\|\boldsymbol{e}^{(0)}\| \leq u(1+\lceil\log_2 n\rceil + O(u))\|C_n^{-1}\|\|\boldsymbol{b}\|. \tag{3.14}$$

The inequality (3.10) now follows from $\|\boldsymbol{b}\| \leq \|C_n\|\|\boldsymbol{x}\|$.

Under the assumption that $\sum_{j=1}^n |\tilde{c}_{ij}b_j| = |\sum_{j=1}^n \tilde{c}_{ij}b_j| = |x_i|$, we obtain (3.8) from (3.13). The inequalities (3.7), (3.9) and (3.11) now follow from (3.6), (3.8) and (3.14), respectively, and the fact that $\|\boldsymbol{r}^{(0)}\| \leq \|C_n\|\|\hat{\boldsymbol{x}}^{(0)} - \boldsymbol{x}\|$. □

Theorem 3.2 shows that the computation of an approximate solution by (2.1) defines a stable method for any nonsingular Cauchy matrix and right-hand side vector. For certain right-hand side vectors and choices of γ, the error in the computed solution is independent of $\kappa(C_n)$, see (3.6) and (3.8). The bounds (3.7) and (3.9) show that the method in these cases is well behaved.

When formula (2.1) is combined with one step of iterative improvement and $\kappa(C_n)$ is bounded appropriately, a well-behaved solution method is obtained. In order to make this statement more precise we introduce some notation. Let $\hat{\boldsymbol{r}}^{(0)} = (\hat{r}_1^{(0)}, \hat{r}_2^{(0)}, \dots, \hat{r}_n^{(0)})^T$ be the *computed* residual vector associated with $\hat{\boldsymbol{x}}^{(0)}$, i.e.,

$$\hat{r}_i^{(0)} = fl(b_i - fl(\sum_{j=1}^n fl(c_{ij}x_j^{(0)}))). \tag{3.15}$$

Let $\hat{\boldsymbol{y}}^{(1)}$ be the computed solution of $C_n\boldsymbol{y} = \hat{\boldsymbol{r}}^{(0)}$ obtained by solution method (2.1), and let $\hat{\boldsymbol{x}}^{(1)} = fl(\hat{\boldsymbol{x}}^{(0)} + \hat{\boldsymbol{y}}^{(1)})$. Also define the (exact) residual vector $\boldsymbol{r}^{(1)} = \boldsymbol{b} - C_n\hat{\boldsymbol{x}}^{(1)}$, and for future reference the corresponding computed residual vector $\hat{\boldsymbol{r}}^{(1)} = (\hat{r}_1^{(1)}, \hat{r}_2^{(1)}, \dots, \hat{r}_n^{(1)})^T$ with components

$$\hat{r}_i^{(1)} = fl(b_i - fl(\sum_{j=1}^n fl(c_{ij}x_j^{(1)}))). \tag{3.16}$$

THEOREM 3.3. *Assume that $\kappa(C_n) \leq 1/u^{1/2}$. Then the computation of $\hat{\boldsymbol{x}}^{(1)}$ in the manner described defines a well-behaved solution method for (1.1). Iterative improvement will in general not reduce the error in the computed approximate solution.*

Proof. We have to show that the residual vector $\boldsymbol{r}^{(1)}$ satisfies an inequality of the form (3.3) with $\hat{\boldsymbol{r}}$ replaced by $\boldsymbol{r}^{(1)}$. This follows by combining Theorem 3.2 with the analysis presented by Jankowski and Wozniakowski [16]. More precisely, Theorem 3.2 and its proof show that the constants q, c_3 and c_4 introduced in [16] are given by

$$\begin{aligned} q &= (\delta_n(1+u)+u)\kappa(C_n), \\ c_3 &= 1, \quad c_4 = 1+\lceil\log_2 n\rceil, \end{aligned} \tag{3.17}$$

where δ_n is defined by (3.12). Following [16], we define the quantities

$$\begin{aligned} \sigma_1 &= (1+q)(1+u)(c_3+(1+c_3u)c_4u\kappa(C_n)+q+(2+q)u, \\ \sigma_2 &= (1+q)(1+u)(1+c_3u)c_4\kappa(C_n)+1. \end{aligned} \tag{3.18}$$

To continue the analysis, we require that $\sigma_1 < 1$. Jankowski and Wozniakowski show that after i steps of iterative improvement, the computed approximate solution $\hat{\boldsymbol{x}}^{(i)}$ satisfies

$$\|\hat{\boldsymbol{x}}^{(i)} - \boldsymbol{x}\| \leq (\sigma_1^i q + (1-\sigma_1)^{-1}\sigma_2 u)\|\boldsymbol{x}\|,$$

where $\boldsymbol{x}$ denotes the (exact) solution of (1.1). Iterative improvement will not significantly reduce the error in the computed approximate solution if

$$q \leq (1-\sigma_1)^{-1}\sigma_2 u. \tag{3.19}$$

We now show that this inequality holds. Ignoring $O(u^2)$-terms, we replace (3.17) by

$$q = u(1+\lceil \log_2 n \rceil)\kappa(C_n) = uc_4\kappa(C_n). \tag{3.20}$$

From (3.18) and (3.20), we obtain

$$\sigma_2 u \geq (1+q)c_4\kappa(C_n)u = (1+q)q,$$

and (3.19) follows. The fact that the computation of $\hat{\boldsymbol{x}}^{(1)}$ defines a well-behaved method now is a consequence of Theorem 4.1 in [16]. Ignoring lower order terms, this theorem and the bounds of Theorem 3.2 show that (3.3) holds for $d_2(n) = 6(1+\lceil \log_2 n \rceil)^2$ with $\hat{\boldsymbol{r}}$ replaced by $\boldsymbol{r}^{(1)}$. □

4. Numerical examples. All computations for Examples 4.1-4.3 below were carried out on a Sun 670 computer in single precision arithmetic, i.e., with $u = 6 \cdot 10^{-8}$, except for the evaluation of a highly accurate approximate solution $\boldsymbol{x}$ in quadruple precision arithmetic. The latter vector was computed by evaluating the entries of the matrix C_n^{-1} and the right-hand side vector $\boldsymbol{b}$ in quadruple precision arithmetic, forming the matrix-vector product in quadruple precision arithmetic and then rounding the resulting vector to single precision accuracy. We will assume that $\boldsymbol{x}$ is the exact solution of (1.1). For future reference, we note that $1/u^{1/2} = 4.1 \cdot 10^3$.

The purpose of this section is to compare the performance of a few numerical methods for the solution of (1.1). The vector $\hat{\boldsymbol{x}}^{(0)}$ denotes an approximate solution computed without iterative improvement and we report the norm of the associated error vector $\hat{\boldsymbol{e}}^{(0)} = fl(\hat{\boldsymbol{x}}^{(0)} - \boldsymbol{x})$. Also, the norm of the computed residual vector $\hat{\boldsymbol{r}}^{(0)}$ defined by (3.15) is displayed. One step of iterative improvement gives the approximate solution $\hat{\boldsymbol{x}}^{(1)}$ and we show the norm of the associated error vector $\hat{\boldsymbol{e}}^{(1)} = fl(\hat{\boldsymbol{x}}^{(1)} - \boldsymbol{x})$, as well as the norm of the residual vector $\hat{\boldsymbol{r}}^{(1)}$ defined by (3.16).

Results for methods based on formula (2.1) with γ chosen to be the median of the entries of the right-hand side vector of the linear system are reported in the tables in the columns labeled `modgast`. When $\gamma = 0$ in (2.1), the inversion formula described by Gastinel [8] is obtained. Results for solution methods based on this inversion formula are reported in the columns labeled `gastinel`.

Examples 4.1-4.3 compare these two schemes with solution methods based on the algorithms `cauchy1` and `cauchy3` proposed by Boros et al. [5]. These algorithms compute LU-factorizations of C_n and C_n^{-1}, respectively, in $O(n^2)$ arithmetic operations. The algorithm `cauchy2`, also presented in [5], is closely related to `cauchy1` and we therefore do not include this algorithm in our comparison.

The linear systems (1.1) used in our numerical experiments are also solved by computing an LU-factorization of C_n by Gaussian elimination with partial pivoting or by computing the Choleski factorization of the matrix. The results are reported in columns labeled `gepp` and `choleski`. We used the LINPACK [7] subroutines SGEFA and SGESL for solution by LU-factorization with partial pivoting, and the LINPACK subroutines SPOFA and SPOSL for solution by Choleski factorization. Both LU and Choleski factorization of an $n \times n$ matrix require $O(n^3)$ arithmetic operations.

Example 4.1. Let C_n be the Hilbert matrix of Example 1.1. This matrix is symmetric and totally positive; see Example 2.2. It is well known that the condition

TABLE 4.1
Entries of right-hand vectors used in the numerical experiments

type	description
1	$b_j = \sqrt{2}$
2	$b_j = \frac{3}{2} - \frac{1}{2}(-1)^j$
3	$b_j = \sqrt{2}(1 + 1/j^2)$
4	$b_j = \sqrt{j}$
5	$b_j = 1/j^2$

TABLE 4.2
Condition numbers of Hilbert matrices

n	$\kappa(C_n)$
3	$7.5 \cdot 10^2$
6	$2.9 \cdot 10^7$
9	$1.1 \cdot 10^{12}$

number $\kappa(C_n)$ grows rapidly with n; see, e.g., [21, 23]. This is illustrated by Table 2. However, the backward error for solution methods based on LU or Choleski factorization of C_n can be bounded independently of $\kappa(C_n)$; see de Boor and Pincus [3]. In order not to destroy the total positivity of C_n, the algorithms `cauchy1` and `cauchy3` are implemented without partial pivoting.

The Tables 3-5 compare the accuracy achieved with the different solution methods for a few values of n and several right-hand side vectors defined by Table 1. Theorem 3.2 shows that the accuracy achieved with the `modgast` method without iterative improvement is not proportional to the condition number of C_n, and Tables 3 and 4 are in agreement with this result. Table 5 shows the performance of the method for a right-hand side vector such that the error bound for the computed solution is proportional to the condition number. Only for $n = 3$ does $\kappa(C_n)$ satisfy the condition of Theorem 3.3. The tables show that generally iterative improvement reduces the residual error, but not the error in the computed solution. In Tables 3-5 the solution method `modgast` without iterative improvement produces the most, or close to the most, accurate approximate solutions. The subroutine SPOFA for Choleski factorization exited with an error flag for the Hilbert matrix of order 9 used for Table 3. □

Boros et al. [5] propose a partial pivoting method for Cauchy matrices based on suitably ordering the nodes s_i before applying a solution method that factors the Cauchy matrix or its inverse. In Examples 4.2-4.3 we use this pivoting method before applying the algorithms `cauchy1` and `cauchy3`. The entries of the right-hand side vector are reordered accordingly. The determination of a suitable ordering of n nodes s_i requires $O(n^2)$ arithmetic operations.

EXAMPLE 4.2. Let C_n be the Toeplitz matrix of Example 1.2 with $a = 1$ and $b = 1/2$. It is known that $\kappa(C_n)$ grows slowly with n, see, e.g., [22]. For instance, $\kappa(C_{20}) = 2.1 \cdot 10^1$ and $\kappa(C_{60}) = 4.7 \cdot 10^1$. Table 6 compares the accuracy achieved when solving a few linear systems of equations. The matrices are very well-conditioned and all solution methods perform well except `cauchy3`, which yields large errors for $n = 60$. □

EXAMPLE 4.3. Let C_n be the Cauchy matrix defined by $s_i = i^{1/2}$ and $t_j = 1/2+j$. Table 7 shows the accuracy achieved for two different right-hand side vectors and $n = 6$. The matrix C_6 is quite ill-conditioned; we have $\kappa(C_6) = 1.5 \cdot 10^6$. The

TABLE 4.3
Hilbert matrix: $s_i = -1 + i$, $t_j = -j$; right-hand side of type 1

n	$\|\boldsymbol{x}\|$	error	`modgast`	`gastinel`	`cauchy1`	`cauchy3`	`choleski`
3	$4.2 \cdot 10^{1}$	$\|\hat{\boldsymbol{e}}^{(0)}\|$	0	$7.6 \cdot 10^{-6}$	$7.6 \cdot 10^{-6}$	$7.6 \cdot 10^{-6}$	$9.2 \cdot 10^{-5}$
		$\|\hat{\boldsymbol{e}}^{(1)}\|$	$2.8 \cdot 10^{-4}$	$8.0 \cdot 10^{-5}$	$3.8 \cdot 10^{-5}$	$6.5 \cdot 10^{-5}$	$9.2 \cdot 10^{-5}$
		$\|\hat{\boldsymbol{r}}^{(0)}\|$	$1.1 \cdot 10^{-6}$	$8.2 \cdot 10^{-6}$	$9.5 \cdot 10^{-7}$	$1.9 \cdot 10^{-6}$	0
		$\|\hat{\boldsymbol{r}}^{(1)}\|$	$6.0 \cdot 10^{-7}$	$6.0 \cdot 10^{-7}$	0	0	0
6	$8.9 \cdot 10^{3}$	$\|\hat{\boldsymbol{e}}^{(0)}\|$	$9.8 \cdot 10^{-4}$	$5.9 \cdot 10^{-1}$	$2.2 \cdot 10^{-1}$	$5.1 \cdot 10^{-2}$	$6.7 \cdot 10^{2}$
		$\|\hat{\boldsymbol{e}}^{(1)}\|$	$6.6 \cdot 10^{-1}$	$3.6 \cdot 10^{2}$	$3.8 \cdot 10^{2}$	$4.5 \cdot 10^{2}$	$1.0 \cdot 10^{2}$
		$\|\hat{\boldsymbol{r}}^{(0)}\|$	$3.2 \cdot 10^{-5}$	$7.0 \cdot 10^{-3}$	$1.2 \cdot 10^{-4}$	$1.5 \cdot 10^{-4}$	$1.2 \cdot 10^{-4}$
		$\|\hat{\boldsymbol{r}}^{(1)}\|$	$9.3 \cdot 10^{-5}$	$1.5 \cdot 10^{-4}$	$6.1 \cdot 10^{-5}$	$1.2 \cdot 10^{-4}$	$1.2 \cdot 10^{-4}$
9	$1.8 \cdot 10^{6}$	$\|\hat{\boldsymbol{e}}^{(0)}\|$	$9.5 \cdot 10^{-7}$	$1.5 \cdot 10^{4}$	$5.3 \cdot 10^{3}$	$5.5 \cdot 10^{3}$	
		$\|\hat{\boldsymbol{e}}^{(1)}\|$	$4.7 \cdot 10^{8}$	$1.2 \cdot 10^{9}$	$8.6 \cdot 10^{8}$	$3.8 \cdot 10^{8}$	
		$\|\hat{\boldsymbol{r}}^{(0)}\|$	$1.9 \cdot 10^{-2}$	$2.2 \cdot 10^{2}$	$1.8 \cdot 10^{-2}$	$3.3 \cdot 10^{-2}$	
		$\|\hat{\boldsymbol{r}}^{(1)}\|$	$8.1 \cdot 10^{0}$	$7.1 \cdot 10^{4}$	$9.0 \cdot 10^{0}$	$5.5 \cdot 10^{0}$	

TABLE 4.4
Hilbert matrix: $s_i = -1 + i$, $t_j = -j$; right-hand side of type 2

n	$\|\boldsymbol{x}\|$	error	`modgast`	`gastinel`	`cauchy1`	`cauchy3`	`choleski`
3	$2.4 \cdot 10^{2}$	$\|\hat{\boldsymbol{e}}^{(0)}\|$	0	0	$1.5 \cdot 10^{-5}$	$3.1 \cdot 10^{-5}$	$7.0 \cdot 10^{-4}$
		$\|\hat{\boldsymbol{e}}^{(1)}\|$	0	0	$1.5 \cdot 10^{-5}$	$8.5 \cdot 10^{-4}$	$1.4 \cdot 10^{-3}$
		$\|\hat{\boldsymbol{r}}^{(0)}\|$	0	0	0	$3.8 \cdot 10^{-6}$	$3.8 \cdot 10^{-6}$
		$\|\hat{\boldsymbol{r}}^{(1)}\|$	0	0	0	$3.8 \cdot 10^{-6}$	0
6	$5.9 \cdot 10^{6}$	$\|\hat{\boldsymbol{e}}^{(0)}\|$	0	$5.0 \cdot 10^{-1}$	$5.0 \cdot 10^{0}$	$2.5 \cdot 10^{-1}$	$5.0 \cdot 10^{5}$
		$\|\hat{\boldsymbol{e}}^{(1)}\|$	$3.4 \cdot 10^{5}$	$3.3 \cdot 10^{5}$	$2.0 \cdot 10^{4}$	$7.3 \cdot 10^{4}$	$3.1 \cdot 10^{5}$
		$\|\hat{\boldsymbol{r}}^{(0)}\|$	$6.3 \cdot 10^{-2}$	$1.3 \cdot 10^{-1}$	$3.1 \cdot 10^{-2}$	$3.1 \cdot 10^{-2}$	$6.3 \cdot 10^{-2}$
		$\|\hat{\boldsymbol{r}}^{(1)}\|$	0	0	$3.1 \cdot 10^{-2}$	$1.6 \cdot 10^{-2}$	$6.3 \cdot 10^{-2}$

methods `modgast`, `gastinel` and `cauchy1` without iterative improvement give the highest accuracy in the computed approximate solution. □

Pairwise summation is optimal in a certain sense, see Vitenko [24], and is well suited for parallel computation. However, there are other summation methods, such as compensated summation due to Kahan, see [15], and doubly compensated summation presented by Priest [18], that can give higher accuracy. In view of that the accuracy achieved when computing the sums (2.1) is important for the performance of the algorithm `modgast`, we will compare several summation methods in the following example.

EXAMPLE 4.4. In this example we replace the pairwise summation of the sums (2.1) used in the method `modgast` in the Examples 4.1 -4.3 by other summation methods. Tables 8 and 9 display the error $\|\hat{\boldsymbol{e}}^{(0)}\|$ in the computed solution obtained by `modgast` when different schemes are used to evaluate the sums in (2.1). Straightforward summation in the order implied by the sums (2.1) is referred to as `ss`, and pairwise summation as `ps`. Kahan's compensated summation (`cs`) seeks to capture the round-off errors in straightforward summation by using auxiliary variables. A detailed description can be found in [15]. Here we only note that the evaluation of a sum with n terms by `cs` requires $4(n-1)$ arithmetic operations. A small forward error in the computed sum can be achieved by using doubly compensated summation (`dcs`) described by Priest [18] if the terms are ordered in decreasing magnitude prior to

TABLE 4.5
Hilbert matrix: $s_i = -1 + i$, $t_j = -j$; right-hand side of type 3

n	$\|\boldsymbol{x}\|$	error	`modgast`	`gastinel`	`cauchy1`	`cauchy3`	`choleski`
3	$4.9 \cdot 10^{1}$	$\|\hat{\boldsymbol{e}}^{(0)}\|$	$3.8 \cdot 10^{-6}$	$2.3 \cdot 10^{-5}$	$1.1 \cdot 10^{-5}$	$7.6 \cdot 10^{-6}$	$1.3 \cdot 10^{-4}$
		$\|\hat{\boldsymbol{e}}^{(1)}\|$	$1.7 \cdot 10^{-4}$	$3.8 \cdot 10^{-6}$	$1.1 \cdot 10^{-4}$	$1.6 \cdot 10^{-4}$	$5.0 \cdot 10^{-5}$
		$\|\hat{\boldsymbol{r}}^{(0)}\|$	$1.2 \cdot 10^{-6}$	$7.9 \cdot 10^{-6}$	$1.9 \cdot 10^{-6}$	$9.5 \cdot 10^{-7}$	$9.5 \cdot 10^{-7}$
		$\|\hat{\boldsymbol{r}}^{(1)}\|$	$1.2 \cdot 10^{-6}$	$9.5 \cdot 10^{-7}$	$1.9 \cdot 10^{-6}$	$9.5 \cdot 10^{-7}$	$9.5 \cdot 10^{-7}$
6	$8.5 \cdot 10^{3}$	$\|\hat{\boldsymbol{e}}^{(0)}\|$	$4.4 \cdot 10^{-1}$	$9.9 \cdot 10^{-1}$	$4.2 \cdot 10^{-1}$	$7.7 \cdot 10^{-1}$	$6.3 \cdot 10^{2}$
		$\|\hat{\boldsymbol{e}}^{(1)}\|$	$2.6 \cdot 10^{2}$	$2.6 \cdot 10^{2}$	$1.4 \cdot 10^{2}$	$4.4 \cdot 10^{2}$	$3.1 \cdot 10^{2}$
		$\|\hat{\boldsymbol{r}}^{(0)}\|$	$3.0 \cdot 10^{-3}$	$1.1 \cdot 10^{-1}$	$6.1 \cdot 10^{-5}$	$1.2 \cdot 10^{-4}$	$1.2 \cdot 10^{-4}$
		$\|\hat{\boldsymbol{r}}^{(1)}\|$	$3.1 \cdot 10^{-4}$	$7.5 \cdot 10^{-3}$	$1.2 \cdot 10^{-4}$	$1.5 \cdot 10^{-4}$	$9.2 \cdot 10^{-5}$

TABLE 4.6
Toeplitz matrix: $s_i = 1/2 + i$, $t_j = j$; right-hand side of type 3

n	$\|\boldsymbol{x}\|$	error	`modgast`	`gastinel`	`cauchy1`	`cauchy3`	`gepp`
20	4.1	$\|\hat{\boldsymbol{e}}^{(0)}\|$	$4.8 \cdot 10^{-7}$	$4.8 \cdot 10^{-7}$	$4.8 \cdot 10^{-7}$	$2.5 \cdot 10^{-6}$	$4.8 \cdot 10^{-7}$
		$\|\hat{\boldsymbol{e}}^{(1)}\|$	$4.8 \cdot 10^{-7}$	$4.8 \cdot 10^{-7}$	$1.2 \cdot 10^{-7}$	$1.2 \cdot 10^{-7}$	$6.0 \cdot 10^{-8}$
		$\|\hat{\boldsymbol{r}}^{(0)}\|$	$4.8 \cdot 10^{-7}$	$6.0 \cdot 10^{-7}$	$1.6 \cdot 10^{-6}$	$9.7 \cdot 10^{-6}$	$9.2 \cdot 10^{-7}$
		$\|\hat{\boldsymbol{r}}^{(1)}\|$	$7.2 \cdot 10^{-7}$	$1.2 \cdot 10^{-6}$	$3.0 \cdot 10^{-7}$	$2.7 \cdot 10^{-7}$	$2.4 \cdot 10^{-7}$
60	6.7	$\|\hat{\boldsymbol{e}}^{(0)}\|$	$9.5 \cdot 10^{-7}$	$9.5 \cdot 10^{-7}$	$4.8 \cdot 10^{-7}$	$5.0 \cdot 10^{1}$	$4.8 \cdot 10^{-7}$
		$\|\hat{\boldsymbol{e}}^{(1)}\|$	$4.8 \cdot 10^{-7}$	$2.4 \cdot 10^{-7}$	$2.4 \cdot 10^{-7}$	$6.4 \cdot 10^{6}$	$2.4 \cdot 10^{-7}$
		$\|\hat{\boldsymbol{r}}^{(0)}\|$	$1.9 \cdot 10^{-6}$	$1.9 \cdot 10^{-6}$	$1.5 \cdot 10^{-6}$	$1.5 \cdot 10^{2}$	$1.2 \cdot 10^{-6}$
		$\|\hat{\boldsymbol{r}}^{(1)}\|$	$3.6 \cdot 10^{-7}$	$1.2 \cdot 10^{-6}$	$8.8 \cdot 10^{-7}$	$2.0 \cdot 10^{7}$	$1.1 \cdot 10^{-6}$

summation. The evaluation of a sum of n terms by `dcs` requires $10(n-1)$ arithmetic operations in addition to the $O(n \log n)$ comparisons for the sorting. `dcs` simulates double precision arithmetic. We also implemented straightforward summation in double precision arithmetic. Error bounds for the latter summation method can be found in [18]. The terms in all sums were computed in single precision arithmetic.

Tables 8 and 9 show that the accuracy achieved by the different summation methods depends both on the matrix and the right-hand side of the linear system. The tables illustrate that straightforward summation in single precision arithmetic can give a significantly larger error than the other summation methods. Pairwise summation is seen to yield high accuracy, as does Kahan's compensated summation. Except for straightforward summation in single precision arithmetic, all summation methods gave roughly the same error. We remark that the accuracy of the approximate solutions computed by `modgast` was nearly the same for all summation methods when applied to the linear systems of Examples 4.1-4.3. The computed examples suggest that the use of double precision arithmetic in the summation or the application of the `dcs` with sorting is, in general, not worthwhile for the present application. □

5. Conclusion. A modification of the inversion formula proposed by Gastinel yields a fast and accurate solution method for linear systems of equations with an $n \times n$ Cauchy matrix. The method makes it possible to determine the condition number in $O(n^2)$ arithmetic operations. The theory indicates, and the computed examples confirm, that iterative improvement should be used only when the condition number is not too large and a small residual error is more important than a small error in the approximate solution. Pairwise summation and compensated summation are good choices of summation methods for the evaluation of the sums in the right-hand side

TABLE 4.7
Cauchy matrix: $s_i = i^{1/2}$, $t_j = 1/2 + j$, $n = 6$

rhs-type	$\|\boldsymbol{x}\|$	error	`modgast`	`gastinel`	`cauchy1`	`cauchy3`	`gepp`
1	$1.4 \cdot 10^2$	$\|\hat{\boldsymbol{e}}^{(0)}\|$	$2.3 \cdot 10^{-5}$	$9.2 \cdot 10^{-4}$	$9.6 \cdot 10^{-4}$	$1.3 \cdot 10^{-2}$	$8.3 \cdot 10^{-2}$
		$\|\hat{\boldsymbol{e}}^{(1)}\|$	$2.1 \cdot 10^{-2}$	$1.2 \cdot 10^{-1}$	$1.3 \cdot 10^{-2}$	$1.1 \cdot 10^{-1}$	$3.1 \cdot 10^{-3}$
		$\|\hat{\boldsymbol{r}}^{(0)}\|$	$1.1 \cdot 10^{-5}$	$1.0 \cdot 10^{-4}$	$9.5 \cdot 10^{-6}$	$5.7 \cdot 10^{-6}$	$1.9 \cdot 10^{-6}$
		$\|\hat{\boldsymbol{r}}^{(1)}\|$	$7.3 \cdot 10^{-6}$	$3.5 \cdot 10^{-6}$	$3.8 \cdot 10^{-6}$	$3.8 \cdot 10^{-6}$	$1.9 \cdot 10^{-6}$
3	$6.8 \cdot 10^3$	$\|\hat{\boldsymbol{e}}^{(0)}\|$	$3.9 \cdot 10^{-3}$	$2.4 \cdot 10^{-3}$	$5.4 \cdot 10^{-3}$	$1.2 \cdot 10^{-2}$	$4.6 \cdot 10^0$
		$\|\hat{\boldsymbol{e}}^{(1)}\|$	$2.3 \cdot 10^0$	$5.1 \cdot 10^{-1}$	$2.9 \cdot 10^0$	$4.0 \cdot 10^0$	$4.3 \cdot 10^0$
		$\|\hat{\boldsymbol{r}}^{(0)}\|$	$6.0 \cdot 10^{-4}$	$7.8 \cdot 10^{-4}$	$2.4 \cdot 10^{-4}$	$1.2 \cdot 10^{-4}$	$1.2 \cdot 10^{-4}$
		$\|\hat{\boldsymbol{r}}^{(1)}\|$	$2.3 \cdot 10^{-4}$	$1.3 \cdot 10^{-4}$	$2.4 \cdot 10^{-4}$	$2.4 \cdot 10^{-4}$	$3.7 \cdot 10^{-4}$

TABLE 4.8
`modgast` *applied to Cauchy systems with Toeplitz matrix:* $s_i = 1/2 + i$, $t_j = j$, $n = 20$

rhs-type	error	single precision arithmetic				double prec. arithm.
		`ss`	`ps`	`cs`	`dcs`	`ss`
3	$\|\hat{\boldsymbol{e}}^{(0)}\|$	$4.8 \cdot 10^{-7}$	$4.8 \cdot 10^{-7}$	$4.8 \cdot 10^{-7}$	$4.8 \cdot 10^{-7}$	$4.8 \cdot 10^{-7}$
4	$\|\hat{\boldsymbol{e}}^{(0)}\|$	$1.4 \cdot 10^{-6}$	$2.4 \cdot 10^{-7}$	$3.6 \cdot 10^{-7}$	$3.6 \cdot 10^{-7}$	$3.6 \cdot 10^{-7}$
5	$\|\hat{\boldsymbol{e}}^{(0)}\|$	$6.0 \cdot 10^{-8}$	$7.5 \cdot 10^{-9}$	$3.0 \cdot 10^{-8}$	$3.0 \cdot 10^{-8}$	$3.0 \cdot 10^{-8}$

of (2.1).

Acknowledgement. We would like to thank Apostolos Gerasoulis and Bill Gragg for comments and reference [18].

REFERENCES

[1] K. E. Batcher, *Sorting networks and their applications*, 1968 Spring Joint Computer Conference, AFIPS Proc. vol. 32, (1968), pp. 307-314.

[2] D. Bini and V. Pan, *Polynomial and Matrix Computations, Vol. 1: Fundamental Algorithms*, Birkhäuser, Boston, 1994.

[3] C. de Boor and A. Pincus, *Backward error analysis of totally positive linear systems*, Numer. Math., 27 (1977), pp. 485-490.

[4] T. Boros, T. Kailath and V. Olshevsky, *Fast algorithms for solving Vandermonde and Chebyshev-Vandermonde systems*, Reprint, Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA, 1994.

[5] T. Boros, T. Kailath and V. Olshevsky, *Fast solution of Cauchy linear systems*, Reprint, Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA, 1995.

[6] P. J. Davis, *Interpolation and Approximation*, Dover, New York, 1975.

[7] J. J. Dongarra, J. R. Bunch, C. B. Moler and G. W. Stewart, *LINPACK Users' Guide*, SIAM, Philadelphia, 1979.

[8] N. Gastinel, *Inversion d'une matrice generalisant la matrice de Hilbert*, Chiffres, 3 (1960), pp. 149-152.

[9] A. Gerasoulis, *A fast algorithm for the multiplication of generalized Hilbert matrices with vectors*, Math. Comp., 50 (1987), pp. 179-188.

[10] A. Gerasoulis, M. D. Grigoriadis and Liping Sun, *A fast algorithm for Trummer's problem*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. s135-s138.

[11] I. Gohberg, T. Kailath and V. Olshevsky, *Fast Gaussian elimination with partial pivoting for matrices with displacement structure*, Math. Comp., 64 (1995), pp. 1557-1576.

[12] I. Gohberg and V. Olshevsky, *Fast algorithms with preprocessing for multiplication of transposed Vandermonde matrix and Cauchy matrix with vector*, J. Complexity, 10 (1994), pp. 411-427.

[13] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys.,

TABLE 4.9
`modgast` *applied to Cauchy systems with Hilbert matrix:* $s_i = -1 + i$, $t_j = -j$, $n = 6$

rhs-type	error	single precision arithmetic				double prec. arithm.
		`ss`	`ps`	`cs`	`dcs`	`ss`
4	$\Vert\hat{\mathbf{e}}^{(0)}\Vert$	$1.5 \cdot 10^{-1}$	$1.4 \cdot 10^{-1}$	$1.4 \cdot 10^{-1}$	$1.4 \cdot 10^{-1}$	$1.4 \cdot 10^{-1}$
5	$\Vert\hat{\mathbf{e}}^{(0)}\Vert$	$1.6 \cdot 10^{-2}$	$1.7 \cdot 10^{-2}$	$1.7 \cdot 10^{-2}$	$1.4 \cdot 10^{-2}$	$1.4 \cdot 10^{-2}$

73 (1987), pp. 325-348.

[14] G. HEINIG, *Inversion of generalized Cauchy matrices and other classes of structured matrices*, in Linear Algebra for Signal Processing, eds. A. Bojanczyk and G. Cybenko, IMA Volume in Mathematics and Its Applications # 69, Springer, New York, 1994, pp. 63-81.

[15] N. J. HIGHAM, *The accuracy of floating point summation*, SIAM J. Sci. Comput., 14 (1993), pp. 783-799.

[16] M. JANKOWSKI AND H. WOZNIAKOWSKI, *Iterative refinement implies numerical stability*, BIT 17 (1977), pp. 303-311.

[17] H. MINC, *Nonnegative Matrices*, Wiley, New York, 1988.

[18] D. M. PRIEST, *On properties of floating point arithmetics: numerical stability and the cost of accurate computations*, Ph.D. thesis, Computer Science Department, University of California at Berkeley, Berkeley, CA, 1992.

[19] L. REICHEL, *Some computational aspects of a method for rational approximation*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 1041-1057.

[20] L. REICHEL, *A matrix problem with application to rapid solution of integral equations*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 263-280.

[21] J. M. TAYLOR, *The condition of Gram matrices and related problems*, Proc. Royal Soc. Edinburgh, 80A (1978), pp. 45-56.

[22] E. E. TYRTYSHNIKOV, *Cauchy and Toeplitz systems*, Linear Algebra Appl., 149 (1991), pp. 1-18.

[23] E. E. TYRTYSHNIKOV, *How bad are Hankel matrices?*, Numer. Math., 67 (1994), pp. 261-269.

[24] I. V. VITENKO, *Optimum algorithms for adding and multiplying on a computer with a floating point*, U.S.S.R. Comput. Math. Math. Phys., 8(5) (1968), pp. 183-195.