Computations with Structured Matrices

Victor Pan[1], Brian Murphy[2], Rhys Eric Rosholt[2]

Department of Mathematics and Computer Science
Lehman College, The City University of New York

[1]Graduate School and University Center
The City University of New York

[2]Doctoral Program in Computer Science
The City University of New York

March 23, 1999

Abstract: We review some basic algorithms for solving problems expressed in the form of matrix equations. The algorithms, which include block Gaussian elimination and recursive block triangular factorization, are first shown for general matrices and then extended to structured matrices, in which case they run much faster and use much less computer memory. In particular, we review the solution of structured linear systems and recall some recent results on the computations with singular structured matrices and on iterative improvement of approximate solutions by using residual correction/Newton's iteration techniques.

Keywords: linear systems of equations, block matrix algorithms, structured matrices, computational complexity

1991 AMS Subject Classification: 65F05, 68Q40, 15A06, 47B35

1. Solving linear systems of equations (some basic classes of algorithms) *Multiplication* of a matrix *by a vector* (MV) and the solution of a *linear system of equations* (LSE) are the most fundamental and most frequent operations in applied linear algebra. For an $m \times n$ matrix $A = (a_{ij})_{i=0,j=0}^{m-1,n-1}$ and two vectors $\mathbf{u} = (u_i)_{i=0}^{m-1}$ and $\mathbf{v} = (v_j)_{j=0}^{n-1}$, one may express these problems by the vector equation,

$$A\mathbf{u} = \mathbf{v}. \tag{1.1}$$

Given A and $\mathbf{u}$ (resp., A and $\mathbf{v}$), the MV (resp., LSE) problem is the problem of computing the vector $\mathbf{v}$ (resp., the vector $\mathbf{u}$).

The straightforward algorithm performs MV by using $(2k-1)l$ flops, a flop being the amount of computer work required on the average per an arithmetic operation, and $(2k-1)l$ is also a lower bound, which shows the optimality of the algorithm.

On the contrary, several competing methods are available for LSE. Hereafter, for simplicity, let the linear system (1.1) be nonsingular, which implies that $m = n$, $\det A \neq 0$, and there exists a unique solution vector $\mathbf{u}$ for given A and $\mathbf{v}$.

The *direct solution methods* compute the vector $\mathbf{u}$ exactly, provided that all flops are performed with infinite precision and no error. The main examples of such methods are *Gaussian elimination* (in several variants, in particular with or without row and/or column interchange, called *pivoting*) and its block version. For a general linear system (1.1) such methods use order of n^3 flops. In practice, the computation is usually performed with rounding to a fixed (single or double) finite precision, and here we enter one of major topics of numerical computing, that is, the estimation of the errors (cf. [GL96], [H96]).

Another major class is *iterative methods*, which produce only approximation to the solution vector $\mathbf{u}$ even if the computations are performed with infinite precision [S96], [G97]. These methods use order of $MA_{\mathbf{v}}$ flops where M is the number of iteration steps required, and $A_{\mathbf{v}}$ flops are required to multiply the input matrix A by a vector. Usually, iterative methods are applied where $M \ll n$ and/or $A_{\mathbf{v}} \ll n^2$. The error analysis of these methods produces estimates for the cumulative errors due to both approximation errors and the errors caused by rounding to a finite precision. There are many iterative methods available [G97], [S95] depending on the class to which the input matrix A belongs.

Some methods (like the *conjugate-gradient* method, for which we will use the abbreviation *c.g.* method) can in principle be used as direct ones (as they always compute $\mathbf{u}$ exactly in n iteration steps assuming infinite precision of computation) but as a rule are used as iterative methods because for many important special classes of matrices A they produce a good approximation to $\mathbf{u}$ even in a few steps performed with finite precision. In practice, the c.g. method is applied with *preconditioning*, that is, applied to a linear system $CA\mathbf{u} = C\mathbf{v}$ equivalent to (1.1), where the nonsingular matrix C is chosen to decrease the number of iteration steps M.

In the next sections, we will give more comments on these groups and special matrices A, with the main goal of studying the case of dense structured matrices A. This will bring us to revisiting the MV and LSE problems, and we will also consider briefly some other linear algebra computations with the latter matrices.

2. Block Gaussian elimination and recursive block triangular factorization Gaussian elimination is the most widely used techniques for the dirtect solution of a linear system of equations (1.1). In practical computations, it is usually performed by computing the LDM^T- or $PLDM^T$- factorization, $A = LDM^T$ or $A = PLDM^T$, where P is a permutation matrix (such that the vectors $\mathbf{w}$ and $P\mathbf{w}$ have the same components written in a distinct order), D is a diagonal matrix, L and M are lower triangular matrices (whose subdiagonal entries are zeros), with W^T here and hereafter denoting the transpose of a matrix or a vector W. We will more closely examine the *block elimination* (with no permutation), applied to a 2×2 block matrix

$$A = \begin{pmatrix} B & C \\ E & G \end{pmatrix}, \tag{2.1}$$

where B is an $(n/2) \times (n/2)$ matrix. (For simplicity, let n be even.) *Block matrix algorithms* are reduced to multiplications and inversions of blocks, which is an advantage, due to efficiency of such matrix operations on modern computers, e.g. loosely-coupled multiprocessors [GL96], [Q94].

Let us assume nonsingular block B in (2.1) and define its *Schur complement* in A,

$$S = G - EB^{-1}C. \tag{2.2}$$

Block Gaussian elimination gives us the following block factorization of A,

$$A = \begin{pmatrix} I & 0 \\ EB^{-1} & I \end{pmatrix} \begin{pmatrix} B & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & B^{-1}C \\ 0 & I \end{pmatrix}. \tag{2.3}$$

Here and hereafter, I and 0 will denote the identity and null matrices of appropriate sizes, $I\mathbf{w} = \mathbf{w}$ for any vector $\mathbf{w}$. It follows from (2.3) that

$$A = \begin{pmatrix} I & -B^{-1}C \\ 0 & I \end{pmatrix} \begin{pmatrix} B^{-1} & 0 \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -EB^{-1} & I \end{pmatrix}. \tag{2.4}$$

Due to (2.4), the computation of the solution $\mathbf{u} = A^{-1}\mathbf{v}$ to equation (1.1) is reduced to the computation of B^{-1} and S and the subsequent multiplications by vectors of the $(n/2) \times (n/2)$ matrices B^{-1} (twice), C, E, and S^{-1}.

We will represent the products EB^{-1} and $B^{-1}C$ by the pairs of matrices E, B^{-1} and B^{-1}, C, partition the matrices B and S into blocks, as we did with A in (2.1), assume nonsingularity of their *leading principal* (that is, northwestern or *l.p.*) submatrices and extend (2.2)-(2.4) recursively. This enables us to compute recursive factorization (RF) of A. The recursion ends where the diagonal blocks become 1×1 matrices. Let us estimate the cost of computing the RF of A and A^{-1}. The computation is reduced to computing a sequence of matrix products and inverses. Let $RF(k)$, $M(k)$, and $I(k)$ denote the number of flops required to compute the recursive factorization, the product and the inverse of $k \times k$ input matrices, respectively. Then (2.1)-(2.4) imply that

$$RF(n) \leq 2RF(n/2) + I(n/2) + 2M(n/2) + n^2, \tag{2.5}$$

$$I(n) \leq 2I(n/2) + 6M(n/2) + O(n^2). \tag{2.6}$$

(2.5),(2.6) reveal that RF recursively reduces the computation of the matrix inverse to a pair of the same problems of half-size, which places RF into the class of *divide-and-conquer algorithms.* We recursively extend (2.6) to the term $2I(n/2)$ and finally (cf. [S96]) arrive at

$$I(n) = O(M(n)). \tag{2.7}$$

Substitute (2.7) into (2.5), recursively obtain similar estimates for RF of B and S and deduce that

$$RF(n) = O(M(n)). \tag{2.8}$$

According to some experimental tests, the described recursive approach to matrix inversion and solving linear systems is highly effective, though it has not been included into current libraries of subroutines for general matrices yet. This should be done sooner or later, actually sooner in the case of structured matrices A, to whose study we will come next.

Remark 2.1 The computation of the RF essentially amounts to recursive inversion of the l.p. blocks and computation of their Schur complements. For a major important class of *real symmetric positive definite (s.p.d.) matrices*, that is, where $A = F^T F$ for some nonsingular F, the matrices A^{-1}, B, B^{-1}, S, S^{-1} are also s.p.d. matrices. Moreover,

$$\text{cond } A = \text{cond } A^{-1} \geq \max\{\text{cond } B, \text{cond } S\} \tag{2.9}$$

in this case. (cond $A = \sigma_1/\sigma_n \geq 1$, $\sigma_1 \geq \cdots \geq \sigma_n > 0$ are the singular values of A; cond A measures sensitivity of the computations with A to the input and rounding errors.) (2.9) shows numerical stability (that is, reasonably low sensitivity to the errors) of the computation of the RF in the case of s.p.d. A. For a nonsingular A, we may shift from (1.1) to the equivalent linear system

$$A^T A\mathbf{u} = A^T \mathbf{v},$$

whose coefficient matrix $A^T A$ is s.p.d.; furthermore, cond $(A^T A) \leq (\text{cond } A)^2$, and in many cases squaring cond A is an unpleasant but still acceptable sacrifice for having an easier life afterwards. Note that $A^{-1} = (A^T A^{-1})A^T$ and $(\det A)^2 = \det(A^T A)$.

3. Dense structured matrices. Definitions and some basic properties

The most celebrated class of structured matrices are Toeplitz and Hankel matrices. They (as well as Cauchy and Vandermonde matrices and various matrices related to these four classes) are omnipresent in many computations for sciences, engineering and electrical engineering.

Table 1. Examples of Matrices with Structure $(i, j = 1, ..., n)$

Toeplitz	Hankel	Cauchy	Vandermonde
$T = [t_{i-j}]$	$H = [h_{i+j}]$	$C = [\frac{1}{x_i - y_j}]$	$V = [x_i^{j-1}]$

The structured matrices of Table 1 are defined by a relatively few parameters (from n to $2n$, versus the n^2 entries representing a general matrix) and can be multiplied by vectors fast, that is, in $O(n \log n)$ flops in Toeplitz and Hankel cases and in $O(n \log^2 n)$ in Cauchy and Vandermonde cases ([BP94], sect. 2.4-2.5). Similar properties hold for the more general classes of Toeplitz-like, Hankel-like, Cauchy-like and Vandermonde-like matrices, defined based on using the associated linear operators of shift (displacement) and/or scaling.

Definition 3.1 $A = (a_{i,j})_{i,j=0}^{n-1}$ is a *Toeplitz matrix* if $a_{i+1,j+1} = a_{i,j}$ for all pairs (i, j), $i, j = 0, \ldots, n-1$. Such a Toeplitz matrix A is called *f-circulant* and is denoted by $C_f(\mathbf{a})$ (for $f \neq 0$ and $\mathbf{a} = (a_{i,0})$) if $a_{i,n-1} = f a_{i+1,0}$, $i = 0, 1, \ldots, n-2$. 1-circulant matrices are called *circulant*. We write $C_f = C_f(\mathbf{i}^{(1)})$, $\mathbf{i}^{(1)} = (0, 1, 0, \ldots, 0)^T$. For $e \neq 0$, $e \neq f$, a pair of $n \times r$ matrices $G = (\mathbf{g}^{(k)})_{k=1}^r$, $H = (\mathbf{h}^{(k)})_{k=1}^r$ is a (C_f, C_e)- *generator of* A of length r if

$$A = C_f(A) + \frac{e}{e-f} \sum_{k=1}^{r} C_f(\mathbf{g}^{(k)}) C_{1/e}(\mathbf{h}^{(k)}) C_f^{-1}, \tag{3.1}$$

$C_f^{-1} = C_{1/f}^T$, and $C_f(A)$ is some f-circulant matrix (sharing its first column with A). The minimum r for which the latter equation holds (for some pair of $n \times r$ matrices G and H) is called the *(C_e, C_f)-rank* of A. For $f = 0$ and $e = \infty$, we define a *(C_0, C_∞)-generator* (G, H) of length r of

$$A = C_0(A) + \sum_{k=1}^{r} C_0(\mathbf{g}^{(k)}) C_0^T (C_0 \mathbf{h}^{(k)}).$$

A is a *Toeplitz-like matrix* if it is given with its (C_e, C_f)-generator (for some fixed e and f) having length $O(1)$ (bounded by a constant). (For two pairs (e, f) and (e_1, f_1), we may easily transform a (C_e, C_f)-generator of A of (3.1) into a (C_{e_1}, C_{f_1})-generator of A of length $O(r)$, so that the definition of Toeplitz-like matrices is invariant in the choice of (e, f). Similar remark applies to the class of Vandermonde-like matrices below.)

Definition 3.2 J is the $n \times n$ *reflection* matrix with 1 on its antidiagonal and 0 elsewhere, $J^2 = I$, $J(\mathbf{v}) = (v_{n-1-i})_{i=0}^{n-1}$ if $\mathbf{v} = (v_i)_{i=0}^{n-1}$. A is a *Hankel* (resp. *Hankel-like*) matrix if AJ is a Toeplitz (resp. Toeplitz-like) matrix.

Definition 3.3 Let $x_0, ..., x_{n-1}$ be n distinct values and let $s_0, t_0, ..., s_{n-1}, t_{n-1}$ be $2n$ distinct values. Let $\mathbf{x} = (x_i)_{i=0}^{n-1}$, $\mathbf{s} = (s_i)_{i=0}^{n-1}$, $\mathbf{t} = (t_j)_{j=0}^{n-1}$. Then the $n \times n$ matrices $C = C(\mathbf{s}, \mathbf{t}) = (\frac{1}{s_i - t_j})_{i,j=0}^{n-1}$ and $V = V(\mathbf{x}) = (x_i^j)_{i,j=0}^{n-1}$ are called *Cauchy* and *Vandermonde* matrices, respectively. Furthermore, the $n \times n$ matrices $C(\mathbf{s}, \mathbf{t}, G, H)$ and $V_f(\mathbf{x}, G, H)$ are said to be given with their $(\mathbf{s}, \mathbf{t})$-*scaling generator* (G, H) and $(f, \mathbf{x})$-*scaling/displacement generator* (G, H), both of length r, respectively, if

$$C(\mathbf{s}, \mathbf{t}, G, H) = \sum_{k=1}^{r} \text{diag}\left(g_i^{(k)}\right)_{i=0}^{n-1} C(\mathbf{s}, \mathbf{t}) \text{diag}\left(h_j^{(k)}\right)_{j=0}^{n-1}, \tag{3.2}$$

$$V_f(\mathbf{x}, G, H) = \sum_{k=1}^{r} \text{diag}\left(\frac{f g_i^{(k)} x_i}{f - x_i^n}\right)_{i=0}^{n-1} V(\mathbf{x}) C_{1/f}^T(\mathbf{h}_j^{(k)}), \tag{3.3}$$

where $\mathbf{g}^{(k)} = (g_i^{(k)})_{i=0}^{n-1}$, $\mathbf{h}^{(k)} = (h_j^{(k)})_{j=0}^{n-1}$, $k = 0, ..., n-1$, G and H are $n \times r$ matrices, $G = (\mathbf{g}^{(k)})_{k=0}^{r-1}$, $H = (\mathbf{h}^{(k)})_{k=0}^{r-1}$, $\text{diag}(w_i)_{i=0}^{n-1}$ is the diagonal matrix with the diagonal entries $w_0, ..., w_{n-1}$, and f is a scalar, $f \neq 0$, $f \neq x_i^n$ for $i = 1, ..., n-1$. The minimum length r in the above representations of the matrices $C(\mathbf{s}, \mathbf{t}, G, H)$ and $V_f(\mathbf{x}, G, H)$ (for all pairs $(\mathbf{s}, \mathbf{t})$ and $(f, \mathbf{x})$, respectively) is called the *scaling rank* and the *scaling/displacement rank* of these matrices, respectively. If $r = O(1)$ as $n \to \infty$, then the above matrices are called *Cauchy-like* and *Vandermonde-like* matrices, respectively.

Since we may multiply an $n \times n$ matrix A by a vector by using $O(n \log n)$ flops where A is a Toeplitz or Hankel matrix and by using $O(n \log^2 n)$ flops where A is a Cauchy or Vandermonde matrix (see e.g. [BP94]), the same asymptotic cost bounds hold for the "like" matrices.

Theorem 3.1 *An $n \times n$ (real or complex) matrix A can be multiplied by a vector by using $M_{\mathbf{v}}(A)$ flops where $M_{\mathbf{v}}(A) = O(n \log n)$ for Toeplitz-like, Hankel-like and Toeplitz-like - Hankel-like matrices A and $M_{\mathbf{v}}(A) = O(n \log^2 n)$ for Cauchy-like and Vandermonde-like matrices A.*

Dealing with structured matrices, we may represent them with their short generators, which decreases the use of memory space and computer time. The generators of the sums and the products of structured matrices are easily expressed via the input generators (cf. [BP94], [GO94]). This is immediately verified for the sums. To obtain the generator for the product, we first recall the following basic results [KKM79], [GO94].

Theorem 3.2 *If A is a Toeplitz-like matrix satisfying (3.1), then*

$$AC_f - C_f A = GH^T. \tag{3.4}$$

Theorem 3.3 *If A is a Cauchy-like matrix satisfying (3.2), then*

$$\mathrm{diag}(\mathbf{s})\ A - A\ \mathrm{diag}(\mathbf{t}) = GH^T. \tag{3.5}$$

Theorem 3.4 *If A is a Vandermonde-like matrix satisfying (3.3), then*

$$(\mathrm{diag}(\mathbf{x}))^{-1}A - AC_f^T = GH^T. \tag{3.6}$$

Theorems 3.2 - 3.4 give us an easy way for computing a generator of small length for a given matrix A, simply by appplying the respective linear operator of the form $AF_1 - F_2A$ and then decomposing its image as a sum of a few outer products of vectors:

$$\Delta_{F_1,F_2}(A) = AF_1 - F_2A = GH^T \tag{3.7}$$

Now we have the following result (cf.[P90], page 193).

Theorem 3.5 *Let K, L, M and N be four fixed matrices and let $F = F_{(K,M)}$, $F_1 = F_{(K,L)}$ and $F_2 = F_{(L,M)}$ be the three operators defined by the following equations:*

$$F_1(A) = KA - AL,\ F_2(B) = LB - BM,\ F(C) = KC - CM.$$

Then the following matrix equations hold:

$$F(AB) = F_1(A)B + AF_2(B).$$

By choosing K, L and M properly, we may obtain short generators of the product of structured matrices. Moreover, we may transform the classes of Vandermonde-like, Cauchy-like and Toeplitz-like matrices into each other, simply by means of their pre- and/or post-multiplication by an appropriate Vandermonde matrix or pair of Vandermonde matrices (cf. [P90] or [BP94], pp. 193-200).

We also have the following result, which bounds the length of the generators of the inverse.

Theorem 3.6 *Let a nonsingular matrix A satisfy (3.7). Then A^{-1} satisfies*

$$\Delta_{F_2,F_1}(A^{-1}) = G_-H_-^T$$

where $G_- = -A^{-1}G$, $H_- = A^{-T}H$, $A^{-T} = (A^{-1})^T = (A^T)^{-1}$.

Proof: Pre- and post-multiply (3.7) by A^{-1}. $\square$

Remark 3.1 Theorem 3.5 defines a short generator for the product constructively, whereas Theorem 3.6 shows the existence of a short generator for the inverse but not in a constructive way (the generator is expressed via the inverse).

4. Solution of structured linear systems Due to Theorem 3.1, iterative methods are highly effective for the solution of structured linear systems (1.1) provided that a few iteration steps suffice to yield a good approximation to the solution. This is the case for a large and important subclass of the class of Toeplitz and Toeplitz-like matrices [KO96], [S99]. In the general case, one may apply direct methods. In particular, the method based on the RF is effective. It runs in $O(n \log^2 n)$ flops and was first proposed for Toeplitz-like matrices in [M74], [M80], [BA80], so we will call it the *MBA algorithm.* In [PZ96] and also [PACPS98], the method was modified for Cauchy-like and Hankel-like cases, and in [OP98] it was extended to a more general class of structured matrices.

The algorithm follows the pattern of section 2, so the entire computation of the RF amounts to the recursive computation of the Schur complements of smaller and smaller sizes, or more precisely, to the computation of their generators. This can be done by using some known explicit expressions.

Theorem 4.1 (see [GO94a], [GO94b], [OP98]) *Let the matrices in $AF^{(1)} - F^{(2)}A = GH^T$ be partitioned as $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$,*

$$F^{(1)} = \begin{bmatrix} F_1^{(1)} & * \\ 0 & F_2^{(1)} \end{bmatrix}, \ F^{(2)} = \begin{bmatrix} F_1^{(2)} & 0 \\ * & F_2^{(2)} \end{bmatrix}, \ G = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix}, \ H = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}. \tag{4.1}$$

*where * stands for some unspecified entries. If A is nonsingular, then the Schur complement $S(A) = A_{22} - A_{21}A_{11}^{-1}A_{12}$ satisfies the equations $S(A)F_2^{(1)} - F_2^{(2)}S(A) = S_G S_H$, $S_G = G_1 - A_{21}A_{11}^{-1}G_2$, $S_H = H_2 - H_1 A_{11}^{-1}A_{12}$.*

The MBA algorithm for a structured matrix A amounts to recursive application of Theorem 4.1. A delicate point is only the Toeplitz-like case. Indeed, unlike (3.5) and (3.6), it does not fit the format of Theorem 4.1 because of the requirements of (4.1) on $F^{(1)}$ and $F^{(2)}$. We, however, rewrite (3.4) for the Hankel-like matrix $\tilde{A} = JA$ and obtain that $J\tilde{A}C_f - C_f J\tilde{A} = GH^T$, $\tilde{A}C_f - JC_f J\tilde{A} = JGH^T$. Then (4.1) holds for $F_1 = C_0$, $F_2 = JC_0J$, and for $\tilde{A}$ and JG replacing A and G, respectively. Thus, we may apply Theorem 4.1 to compute the RF of the Hankel-like matrix $\tilde{A} = JA$. Having $\tilde{A}^{-1}$ and/or $\det \tilde{A}$, we immediately recover $A^{-1} = \tilde{A}^{-1}J$ and $\det A = (\det \tilde{A}) \det J = (-1)^{n-1} \det \tilde{A}$.

By combining Theorems 3.1 and 4.1 at every step of recursive computation of the Schur complements, we finally compute the RF of A, as well as a generator of A^{-1} and $\det A$ by using $O(n \log^2 n)$ flops for a Toeplitz-like/Hankel-like matrix A and $O(n \log^3 n)$ for the Cauchy-like or Vandermonde-like matrix A.

Based on Theorem 3.5, we may (as we mentioned already) transform one class of structured matrices into another, in particular, Cauchy-like and Vandermonde-like matrices into Toeplitz-like matrices (cf. [P90] or [BP94]). The cost of such a transformation is $O(n \log^2 n)$ flops, and therefore, any algorithm for solving LSE and computing the inverse and the determinant is also transformed at this cost. Since this also bounds the solution cost in the Toeplitz-like case, the overall asymptotic solution cost in the Cauchy-like and Vandermonde-like cases also decreases to $O(n \log^2 n)$.

The recipe of Remark 2.1 for avoiding degeneracy cannot be applied explicitly to Cauchy-like matrices A because the condition of the separation of vectors $\mathbf{s}$ and $\mathbf{t}$ of Definition 3.3 is violated for $A^T A$ in this case, but the recipe can be applied implicitly (see [PZ96], [PACPZ99]).

For practical numerical computations with rounding, the absence of pivoting in the RF algorithm may be a weak point. To avoid such a problem, one may apply the algorithm of [GKO95] based on a modification of [H95], where the transformation of [P90] from Toeplitz-like into Cauchy-like matrices was replaced by a much simpler one (which essentially amounts to Fast Fourier Transform, FFT), and then usual Gaussian elimination with pivoting can be performed for the resulting Cauchy-like matrices in $O(n^2)$ flops (still much better than order of n^3 for general matrices).

5. Further extensions; computations with singluar matrices and iterative improvement of approximate solution There are several directions for the extension of the presented algorithms and techniques, which can be found in [BP94], [K95], [PZ96], [OP98], [PBRZ99], [PZACP99] and the bibliography therein. In particular, in [K95], [PZ96], [PZACP99], the recursive factorization approach was extended to fast solution of linear systems (1.1) that are consistent, that is, have solution vectors $\mathbf{u}$, but singular, that is, $\det A = 0$. Furthermore, as a by-product, the algorithms also compute rapidly the rank of a singular structured matrix and a basis for its null space. The algorithms of [K95], [PZ96], [PZACP99] are sensitive to roundoff errors and are designed for computations in finite fields $\mathbf{F}$, such as the fields of integers modulo a fixed prime, where the entries of A, $\mathbf{u}$, $\mathbf{v}$ are in $\mathbf{F}$ and all computations are performed in $\mathbf{F}$. The algorithms use randomization to avoid singularities at the auxiliary steps.

Iterative methods do not apply in finite fields, of course, but there are other important extensions of our results. In particular, the conjugate gradient (c.g.) algorithms converge to the solution of LSE linearly, that is, the number of correct output bits is proportional to the number of iteration steps. In [PBRZ99] some iterative algorithms are covered that start with a sufficiently close but still rough approximation to the solution and improve it quadratically by squaring the norm, $\mathbf{e}_i$, of the approximation error vectors in each i-th iteration step, that is, we have $\mathbf{e}_{i+1} = O(\mathbf{e}_i^2)$). In other words, the algorithms roughly double the number of correct bits in every iteration step. The algorithms develop the Newton's iteration/residual correction methods, which are also known as useful techniques for the numerical computations with roundoff for singular matrices [PS91], the singular value decomposition [GL96] being an effective alternative technique. In [PBRZ99] these methods are adjusted to structured matrix computations based on the known inversion formulas for Toeplitz matrices and the displacement representations of Toeplitz-like matrices. In this case, each iteration step is performed in $O(n \log n)$ time, and thus the algorithms produce b

correct output bits in $O(\log b)$ iterations. The algorithms can be extended to the case of structured matrices of other classes.

References

[BA80] R.R.Bitmead, B.D.O.Anderson, Asymptotically Fast solution of Toeplitz and Related Systems of Linear Equations, *Linear Algebra Appl.*, 34, 103-116, 1980.

[BP94] D.Bini, V.Y.Pan, *Polynomial and Matrix Computations, Volume 1: Fundamental Algorithms*, Birkhäuser, Boston, 1994.

[G97] A.Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM Publications, Philadelphia, 1997.

[GKO95] I.Gohberg, T.Kailath, V.Olshevsky, Gaussian Elimination with Partial Pivoting for Structured Matrices, *Math. of Computation*, 64, 1557-1576, 1995.

[GL96] G.H.Golub, C.F.Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, 1996 (third edition).

[GO94] I.Gohberg, V.Olshevsky, Complexity of Multiplication with Vectors for Structured Matrices, *Linear Algebra Appl.*, 202, 163-192 1994.

[GO94a] I.Gohberg, V.Olshevsky, Fast Algorithm for Matrix Nehari Problem, *Proc. MTNS-93, Systems and Networks: Mathematical Theory and Applications*, v.2, Invited and Contributed Papers (edited by U.Helmke, R.Mennicken, J.Sauers), Academy Verlag, 687-690, 1994.

[GO94b] I.Gohberg, V.Olshevsky, Fast State Space Algorithms for Matrix Nehari and Nehari-Takagi Interpolation Problems, *Integral Equations and Operator Theory*, 20, 1, 44-83, 1994.

[GO96] I.Gohberg, V.Olshevsky, Fast inversion of Vandermonde and Vandermonde-like Matrices, in *Communications, Computation, Control and Signal Processing: A Tribute to Thomas Kailath* (Eds. A.Pauiraj, V.Roychowdhury and C.Shaper), 205-221, Kluwer Academic Publishing, 1996.

[H96] N.J.Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, 1996.

[K95] E.Kaltofen, Analysis of Coppersmith's Block Wiedemann Algorithm for the Parallel Solution of Sparse Linear Systems, *Math. of Computation*, 64, 210, 777-806, 1995.

[KKM79] T.Kailath, S.-Y.Kung, M.Morf, Displacement Ranks of Matrices and Linear Equations, *J. Math. Anal. Appl.*, 68, 2, 395-407, 1979.

[M80] M.Morf, Doubling Algorithms for Toeplitz and Related Equations, *Proc. IEEE Internat. Conf. on ASSP*, 954-959, IEEE Computer Society Press, 1980.

[OP98] V.Olshevsky, V.Y.Pan, A Unified Superfast Algorithm for Boundary Rational Tangential Interpolation Problem and for Inversion and Factorization of Dense Structured Matrices, *Proc. 39th Annual IEEE Symposium on Foundation of Computer Science*, 192-201, IEEE Computer Society Press, 1998.

[P90] V.Y.Pan, Computations with Dense Structured Matrices, *Math. of Computation*, 55, 191, 179-190, 1990.

[PACPS98] V.Y.Pan, M.Abu Tabanjeh, Z.Chen, S.Providence, A.Sadikou, Transformations of Cauchy Matrices for Trummer's Problem and a Cauchy-like Linear Solver, *Proc. of 5th Annual International Symposium on Solving Irregularly Structured Problems in Parallel (Irregular 98)*, (A.Ferreira, J.Rolim, H.Simon, S.-H.Teng, Editors), Lecture Notes in Computer Science, 1457, 274-284, Springer, 1998.

[PBRZ99] V.Y.Pan, S.Branham, R.E.Rosholt, A.Zheng, Newton's Iteration for Structured Matrices and Linear Systems of Equations, chapter in *Fast Reliable Algorithms for Matrices with Structure* (eds. T.Kailath, A.Sayed), SIAM, 1999.

[PS91] V.Y.Pan, R.Schreiber, An Improved Newton Iteration for the Generalized Inverse of a Matrix, with Applications, *SIAM J. Sci. Stat. Comput.*, 12, 5, 1109-1131, 1991.

[PZ96] V.Y.Pan, A.Zheng, Fast Algorithms for Cauchy-like Matrix Computations and an Extension", accepted by *Linear Algebra and Its Applications.*

[PZACP99] V.Y.Pan, A.Zheng, M.Abu Tabanjeh, Z.Chen, S.Providence, Superfast Computations with Singular Structured Matrices over Abstract Fields, *Proc. of Second Workshop on Computer Algebra in Scientific Computing (CASC99)*, Springer, 1999.

[Q94] M.J.Quinn, *Parallel Computing: Theory and Practice*, McGraw-Hill, New York, 1994.

[S69] V.Strassen, Gaussian Elimination is Not Optimal, *Numer. Math.*, 13, 354-356, 1969.

[S96] Y.Saad, *Iterative methods for Sparse Linear Systems*, PWS Publishing Co. Boston, 1996.

[S99] G. Strang, Discrete Cosine Transform, *SIAM Review*, 41, 1999.