MATHEMATICS OF COMPUTATION
VOLUME 50, NUMBER 181
JANUARY 1988, PAGES 179–188

A Fast Algorithm for the Multiplication of Generalized Hilbert Matrices with Vectors*

By A. Gerasoulis

Abstract. The existence of a fast algorithm with an $O_A(n(\log n)^2)$ time complexity for the multiplication of generalized Hilbert matrices with vectors is shown. These matrices are defined by $(B_p)_{i,j} = 1/(t_i - s_j)^p$, $i,j = 1,\ldots,n$, $p = 1,\ldots,q$, $q \ll n$, where t_i and s_i are distinct points in the complex plane and $t_i \neq s_j$, $i,j = 1,\ldots,n$. The major contribution of the paper is the stable implementation of the algorithm for two important sets of points, the Chebyshev points and the nth roots of unity. Such points have applications in the numerical approximation of Cauchy singular integral equations. The time complexity of the algorithm, for these special sets of points, reduces to $O_A(n \log n)$.

1. Introduction. Let us define the generalized Hilbert matrices $\mathbf{B}_p$, $p = 1,\ldots,q$, $q \ll n$, by

$$(B_p)_{i,j} = \frac{1}{(t_i - s_j)^p}, \qquad i,j = 1,\ldots,n, \tag{1}$$

where t_i and s_i are distinct points in the complex plane and $t_i \neq s_j$, $i,j = 1,\ldots,n$. The special case, $p = 1$, $t_i = i$ and $s_j = -j + 1$, is the well-known Hilbert matrix

$$(B_1)_{i,j} = \frac{1}{i + j - 1}, \qquad i,j = 1,\ldots,n. \tag{2}$$

For the case $p = 1$ and $s_j = t_j =: c_j$, we define the matrix

$$T_{i,j} = \begin{cases} \dfrac{1}{c_i - c_j} & \text{if } i \neq j, \\ 0 & \text{if } i = j, \end{cases} \qquad i,j = 1,\ldots,n. \tag{3}$$

In [11], the following question was considered:

"Given a vector $\mathbf{y}$. Does there exist an algorithm for computing the product $\mathbf{Ty}$ in less than $O_A(n^2)$ operations?"

The time or space complexity $O_A(f(n))$ of a straight-line model of computation is defined in Aho et al. [1, pp. 19–22]. This problem was initially posed by Golub in [12] and it is known as Trummer's problem. It has generated considerable interest because of various applications in the computation of conformal mappings (Trummer [22]), the zeta function (Odlyzko and Schönhage [18]), the numerical evaluation of singular integrals (this paper and Reichel [19]) and particle simulations (Greengard and Rokhlin [13] and Rokhlin [20]).

Received March 18, 1986; revised September 5, 1986.

1980 *Mathematics Subject Classification* (1985 *Revision*). Primary 68Q25, 65F30, 65R20, 65D30.

Key words and phrases. Generalized Hilbert matrices, fast algorithms, computational complexity, singular integrals.

*This material is based upon work supported by the National Science Foundation under Grant No. DMS-8506464 and DMS-8706122.

In [11] we have proposed an $O_A(n(\log n)^2)$ algorithm for Trummer's problem, henceforth the GGS algorithm. The GGS algorithm uses Fast Fourier Transform (FFT) polynomial multiplication, polynomial interpolation and polynomial evaluation at n distinct points. In Section 2 we show that the GGS algorithm can be extended to include the matrices defined in (1). The time complexity of the extended algorithm is the same as the GGS. In Section 3 we present examples of generalized Hilbert matrices arising in the numerical approximation of singular integrals. In Sections 4 and 5 we present a stable implementation of the algorithm for the Chebyshev points $s_j = \cos(j\pi/n)$, $j = 1, \ldots, n-1$ and $t_k = \cos((2k-1)\pi/2n)$, $k = 1, \ldots, n$, and for the nth roots of unity $c_k = \exp(2\pi i(k-1)/n)$, $i = \sqrt{-1}$, $k = 1, \ldots, n$. Both sets of points have important applications in the numerical approximation of Cauchy singular integrals. The time complexity of the algorithm for these special sets of points reduces to $O_A(n \log n)$. In Section 5 we also discuss two approximate algorithms, for specially distributed sets of points, recently proposed by Greengard and Rokhlin [13], Rokhlin [20], Odlyzko and Schönhage [18] and Reichel [19].

2. An Extended GGS Algorithm. In this section we briefly describe an extension to the GGS algorithm for the multiplication of $\mathbf{B}_p$ with a vector. We notice that the problem of multiplying $\mathbf{B}_p\mathbf{y}$ is equivalent to evaluating the function $f_p(x)$ at the points t_i, $i = 1, \ldots, n$, where

$$f_p(x) = \sum_{j=1}^{n} \frac{y_j}{(x - s_j)^p}, \qquad p = 1, \ldots, q. \tag{4}$$

We consider $f_1(x)$ first. Following Gastinel [7], we express $f_1(x)$ as the ratio of two polynomials $h(x)$ and $g(x)$, where $g(x)$ is an nth degree polynomial defined by

$$g(x) = \prod_{j=1}^{n} (x - s_j) \tag{5}$$

and $h(x)$ is a polynomial, of degree at most $n-1$, determined from

$$f_1(x) = \frac{h(x)}{g(x)} = \sum_{j=1}^{n} \frac{y_j}{x - s_j}. \tag{6}$$

Setting $x = s_i$, $i = 1, \ldots, n$, in (6), we derive

$$h(s_i) = y_i g'(s_i), \qquad i = 1, \ldots, n, \tag{7}$$

which implies that $h(x)$ is the interpolation polynomial for the points $(s_i, y_i g'(s_i))$, $i = 1, \ldots, n$.

Observe that $f_p(x) = f'_{p-1}(x)/(1-p)$, $p = 2, 3, \ldots, q$. Therefore, expressions for the computation of $\mathbf{B}_p\mathbf{y}$ can be derived by using (6). For example, for $p = 1, 2$, we have that

$$f_1(t_i) = \frac{h(t_i)}{g(t_i)}, \quad f_2(t_i) = -f_1'(t_i) = -\frac{h'(t_i)}{g(t_i)} + \frac{h(t_i)g'(t_i)}{g^2(t_i)}, \qquad i = 1, \ldots, n. \tag{8}$$

Similarly, Trummer's problem $\mathbf{x} = \mathbf{T}\mathbf{y}$ is equivalent to evaluating (Gerasoulis et al. [11])

$$x_j = \frac{h'(c_j) - \frac{1}{2}y_j g''(c_j)}{g'(c_j)}, \qquad j = 1, \ldots, n. \tag{9}$$

We now present an efficient algorithm for the evaluation of $f_1(t_i)$ and $f_2(t_i)$, $i = 1, \ldots, n$. The algorithm can be easily extended to include $p \geq 3$.

Procedure FAST$(n, \mathbf{t}, \mathbf{s}, \mathbf{y})$; **return** $\mathbf{f}_1, \mathbf{f}_2$;

1. Compute the coefficients of $g(x)$ in its power form, by using FFT polynomial multiplication, in $O_A(n(\log n)^2)$ time (e.g. Horowitz [15], Aho et al. [1, Theorem 8.14, p. 299], Henrici [14, pp, 36–38]);
2. Compute the coefficients of $g'(x)$ in $O_A(n)$ time;
3. Evaluate $g(t_i)$, $g'(t_i)$, $i = 1, \ldots, n$, and $g'(s_j)$, $j = 1, \ldots, n$, in $O_A(n(\log n)^2)$ time (Aho et al. [1, Corollary 2, p. 294]);
4. Compute $h_j = y_j g'(t_j)$, $j = 1, \ldots, n$, in $O_A(n)$ time;
5. Find the interpolation polynomial $h(x)$ for the points (s_j, h_j), $j = 1, \ldots, n$, in $O_A(n(\log n)^2)$ time (Aho et al. [1, Theorem 8.14, p. 299]);
6. Compute the coefficients of $h'(x)$ in $O_A(n)$ time, and evaluate $h(t_i)$ and $h'(t_i)$, $i = 1, \ldots, n$, in $O_A(n(\log n)^2)$ time, following the same technique as in steps 2 and 3;
7. Compute $f_1(t_i) = h(t_i)/g(t_i)$ and $f_2(t_i) = -h'(t_i)/g(t_i) + h(t_i)g'(t_i)/g^2(t_i)$, $i = 1, \ldots, n$, in $O_A(n)$ time;

end FAST;

The space and time complexity of FAST are $O_A(n \log n)$ and $O_A(n(\log n)^2)$, respectively. In Sections 4 and 5, we consider two important special cases for which the time complexity reduces to $O_A(n \log n)$.

3. Generalized Hilbert Matrices. In this section we present generalized Hilbert matrices which arise in the quadrature approximation of Cauchy singular integrals. We consider the Cauchy principal value integral

$$(10) \qquad I(y; s) = \frac{1}{\pi} \int\!\!\!\!\!\!-_{-1}^{1} w(t) \frac{y(t)}{t - s}\, dt, \qquad |s| < 1,\ w(t) = (1 - t)^{\alpha}(1 + t)^{\beta},$$

where α and β are constants and $-1 < \alpha, \beta < 1$. Hadamard's finite part integral is defined by

$$(11) \qquad I_H(y; s) = \frac{dI(y; s)}{ds} = \frac{1}{\pi} \int\!\!\!\!\!\!=_{-1}^{1} w(t) \frac{y(t)}{(t - s)^2}\, dt, \qquad |s| < 1,$$

where it is assumed that $w(t)$ and $y(t)$ are such that the derivative of $I(y; s)$ exists. The singular integrals (10) and (11) arise in fields such as aerodynamics, waveguide theory, scattering, fracture mechanics and others. A particular example is the equation

$$(12) \qquad \frac{1}{\pi} \int\!\!\!\!\!\!-_{-1}^{1} (1 - t^2)^{-1/2} \frac{y(t)}{t - s}\, dt = f(s), \qquad |s| < 1,$$

which arises in fracture mechanics. The solution $y(t)$ represents the derivative of the crack opening under a given pressure distribution $f(s)$ along $(-1, 1)$ (Erdogan and Gupta [6], Kaya [17]).

We will derive quadratures for the singular integral (10). These quadratures give rise to matrices $\mathbf{B}_1$ and $\mathbf{T}$. Quadratures for $I_H(y; s)$ and the matrix $\mathbf{B}_2$ can be

obtained from the quadratures for $I(y;s)$ via (11). By rewriting (10) as

$$I(y;s) = \frac{1}{\pi}\int_{-1}^{1} w(t)\frac{y(t)-y(s)}{t-s}\,dt + \frac{y(s)}{\pi}\fint_{-1}^{1}\frac{w(t)}{t-s}\,dt \tag{13}$$

we see that classical quadratures may be used to approximate $I(y;s)$. The second integral in (13) can be computed once to within any given tolerance by using power series expansions and contour integration (Gautschi and Wimp [8]) and the first integral via a quadrature for different functions $y(t)$. FAST could be used to speed up the quadrature computations.

We consider only two cases of the weight $w(t)$. The analysis can easily be extended to the general weight function $w(t)$ in (10).

The case $\alpha = \beta = 0$. Here, the weight function is $w(t) = 1$. Applying the trapezoidal rule in (13), we derive

$$I_n(y;s) = \sum_{i=0}^{n+1} w_i\frac{y(t_i)-y(s)}{t_i-s} + \frac{y(s)}{\pi}\ln\frac{|1-s|}{|1+s|}, \qquad |s|<1, \tag{14}$$

where $t_i = -1 + ih$, $i = 0,1,\ldots,n+1$, $h = 2/(n+1)$, $w_0 = w_{n+1} = h/(2\pi)$ and $w_i = h/\pi$, $i = 1,\ldots,n$. Trummer's matrix $\mathbf{T}$ can be derived by setting $s = t_j$, $j = 1,\ldots,n$, in (14):

$$I_n(y;t_j) = \sum_{\substack{i=1\\ i\neq j}}^{n}\frac{w_i y(t_i)}{t_i-t_j} - y(t_j)\sum_{\substack{i=1\\ i\neq j}}^{n}\frac{w_i}{t_i-t_j} + F(t_j), \qquad j = 1,\ldots,n,$$

$$F(t_j) = \frac{w_0(y(t_0)-y(t_j))}{t_0-t_j} + \frac{w_{n+1}(y(t_{n+1})-y(t_j))}{t_{n+1}-t_j} + w_j y'(t_j) + \frac{y(t_j)}{\pi}\ln\frac{|1-t_j|}{|1+t_j|}, \tag{15}$$

where we have assumed that $y'(t_j)$ exists. The computation of $y'(t_j)$ can be performed via a finite difference formula with the same or better accuracy than (14).

The case $\alpha = \beta = -1/2$. For this special case the weight function becomes $w(t) = (1-t^2)^{-1/2}$. Using the Gauss-Chebyshev quadrature and the identities (Erdogan and Gupta [6])

$$n^{-1}\sum_{i=1}^{n}\frac{1}{t_i-s} = -\frac{U_{n-1}(s)}{T_n(s)}, \qquad \fint_{-1}^{1}\frac{(1-t^2)^{-1/2}}{t-s}\,dt = 0, \qquad |s|<1, \tag{16}$$

we see that $I(y;s)$ can be approximated by

$$I_n(y;s) = n^{-1}\sum_{i=1}^{n}\frac{y(t_i)-y(s)}{t_i-s} = n^{-1}\sum_{i=1}^{n}\frac{y(t_i)}{t_i-s} + \frac{U_{n-1}(s)}{T_n(s)}y(s), \tag{17}$$

where $t_i = \cos((2i-1)\pi/2n)$, $i = 1,\ldots,n$, are the zeros of the Chebyshev polynomial $T_n(t)$.

Setting $s = s_j =: \cos(j\pi/n)$, $j = 1,\ldots,n-1$, in (17), where s_j are the zeros of the Chebyshev polynomial of the second kind $U_{n-1}(s)$, we derive summations similar to $\mathbf{B}_1\mathbf{y}$:

$$I_n(y;s_j) = n^{-1}\sum_{i=1}^{n}\frac{y(t_i)}{t_i-s_j}, \qquad j = 1,\ldots,n-1, \tag{18}$$

while setting $s = t_i$, $i = 1,\ldots,n$, we derive $\mathbf{Ty}$.

4. An Application of FAST. In this section we present an $O_A(n \log n)$ stable implementation of FAST for the Chebyshev points. We consider the numerical solution of (12). Using (18) to approximate (12), we obtain the $(n-1)\times n$ algebraic system

$$\mathbf{Ay} = \mathbf{f}, \qquad A_{j,i} = \frac{1}{n(t_i - s_j)}, \qquad j = 1,\dots,n-1,\ i = 1,\dots,n, \tag{19}$$

where $\mathbf{y} = [y(t_1), y(t_2),\dots,y(t_n)]^T$ and $\mathbf{f} = [f(s_1), f(s_2),\dots,f(s_{n-1})]^T$. By using identity (2.4) of [9], we can easily derive the following right inverse $\mathbf{A}^I$ of $\mathbf{A}$, and consequently obtain the solution of (19) from

$$\mathbf{y} = \mathbf{A}^I\mathbf{f} + b_n\mathbf{u}_n, \qquad (A^I)_{i,j} = \frac{1 - s_j^2}{n(t_i - s_j)}, \qquad i = 1,\dots,n,\ j = 1,\dots,n-1, \tag{20}$$

where b_n is an arbitrary constant and $\mathbf{u}_n$ is a vector with all elements equal to one. In the following example, we apply FAST to the computation of

$$(\mathbf{A}^I\mathbf{f})_i = n^{-1}\sum_{j=1}^{n-1}\frac{(1 - s_j^2)f(s_j)}{t_i - s_j}, \qquad i = 1,\dots,n. \tag{21}$$

Example 1. The algorithm presented below is a modification of FAST for the INPUT: $t_i = \cos((2i-1)\pi/2n)$, $i = 1,\dots,n$, $s_j = \cos(j\pi/n)$and $y_j = f(s_j)$, $j = 1,\dots,n-1$.

1. Instead of computing $g(x)$ in its power form, we will use an analytic expression. We have

$$g(x) = \prod_{j=1}^{n-1}(x - s_j) = d_n U_{n-1}(x) = d_n \sin(n\theta)/\sin(\theta),$$

where $d_n = 2^{-(n-1)}$ and $\cos(\theta) = x$.

2. Similarly, $g'(x) = -d_n[nT_n(x) - xU_{n-1}(x)]/(1 - x^2)$.

3. Since $g'(s_j) = d_n(-1)^{j+1}n/(1 - s_j^2)$, the complexity for this step reduces to $O_A(n)$ time.

4. Equation (7) and step 3 above imply that

$$h(s_j) = d_n(-1)^{j+1}y_j, \qquad j = 1,\dots,n-1.$$

5. We find $h(x)$ by using orthogonal polynomial interpolation. We first set

$$h(x) = \sum_{k=1}^{n-1} a_k U_{k-1}(x) \tag{22}$$

and use the fact that Gaussian quadratures for $n-1$ points are exact for polynomials of degree $\le 2n-3$ to derive the following orthogonality relation,

$$\begin{aligned}&\int_{-1}^{1}(1 - x^2)^{1/2}U_l(x)U_m(x)\,dx\\ &\quad = \pi n^{-1}\sum_{k=1}^{n-1}(1 - s_k^2)U_l(s_k)U_m(s_k) = \begin{cases}\pi & \text{if } l = m,\\ \frac{\pi}{2} & \text{if } l \ne m,\\ 0 & \end{cases}\end{aligned} \tag{23}$$

where $l+m \leq 2n-3$. From the last two equations we see that for $k = 1, \ldots, n-1$,

$$(24)\quad a_k = 2n^{-1}\sum_{j=1}^{n-1}(1-s_j^2)h(s_j)U_{k-1}(s_j) = \sum_{j=1}^{n-1} 2x_j \sin\left(k\frac{j\pi}{n}\right),$$

$$x_j = n^{-1}h(s_j)\sin\left(\frac{j\pi}{n}\right).$$

Therefore, a_k can be computed via FFT in $O_A(n\log n)$ time (Aho et al. [1]).

6. From step 1 and (22), we see that for $i = 1, \ldots, n$,

$$(25)\quad g(t_i) = \frac{d_n(-1)^{i+1}}{\sin((2i-1)\pi/2n)},$$

$$h(t_i) = \sum_{k=1}^{n-1} a_k \sin\left(k\frac{(2i-1)\pi}{2n}\right) \Big/ \sin\left(\frac{(2i-1)\pi}{2n}\right).$$

7. Finally, $\mathbf{A}^I\mathbf{f}$ is computed from (26) by using FFT in $O_A(n\log n)$ time,

$$(26)\quad f_1(t_i) = \frac{h(t_i)}{g(t_i)} = \frac{1}{d_n}\sum_{k=1}^{n-1}(-1)^{i+1}a_k \sin\left(k\frac{(2i-1)\pi}{2n}\right), \qquad i = 1, \ldots, n.$$

The total cost of the above implementation of FAST is $O_A(n\log n)$, since it only requires the application of FFT twice. In Table 4.1 we present our computational experience with FAST, and with the conventional algorithm for the multiplication of a matrix with a vector directly, which requires $O_A(n^2)$ operations. We have chosen the function $f(s) = 1$. In this case, the summations can be obtained exactly from the identity (e.g. Erdogan and Gupta [6])

$$(27)\quad n^{-1}\sum_{j=1}^{n-1}\frac{1-s_j^2}{t_i - s_j} = t_i, \qquad i = 1, \ldots, n.$$

TABLE 4.1

The performance of FAST versus direct multiplication for $f(s) = 1$.

$O_A(n\log n)$ Algorithm				$O_A(n^2)$ Algorithm	
k	$n = 2^k$	Time in sec.	Max. Error	Time in sec.	Max. Error
2	4	0.0013	0.372×10^{-7}	0.0004	0.223×10^{-7}
3	8	0.0031	0.447×10^{-7}	0.0014	0.968×10^{-7}
4	16	0.0054	0.447×10^{-7}	0.0055	0.194×10^{-6}
5	32	0.0122	0.671×10^{-7}	0.0222	0.484×10^{-6}
6	64	0.0248	0.596×10^{-7}	0.0889	0.789×10^{-6}
7	128	0.0524	0.104×10^{-6}	0.3563	0.200×10^{-5}
8	256	0.1082	0.372×10^{-6}	1.4264	0.400×10^{-5}
9	512	0.2335	0.738×10^{-6}	5.7057	0.814×10^{-5}
10	1024	0.4831	0.114×10^{-5}	22.8265	0.167×10^{-4}
11	2048	0.9975	0.627×10^{-5}	91.3278	0.336×10^{-4}
12	4096	2.0592	0.743×10^{-5}	365.2306	0.713×10^{-4}

The maximum error shown in Table 4.1 is obtained from $\max_i |t_i - f_1(t_i)|$ in the fourth column, and from $\max_i |t_i - (\mathbf{B}_1\mathbf{y})_i|$ in the last column. The computations have been performed on a DEC-20 by using the single-precision subroutines

SINT and SINQF from FFTPACK of NETLIB (Swarztrauber [21]), which are most efficient when n is a power of 2. The table shows that FAST outperforms the conventional algorithm for all $n \geq 16$ and that it also attains a better accuracy. Similar results have been obtained for several other choices of $f(s)$. □

The algorithm described in the above example could be used in the solution of the singular integral equation

$$\frac{1}{\pi} \int_{-1}^{1} w(t) \frac{y(t)}{t-s}\, dt + \lambda \int_{-1}^{1} w(t) K(t,s) y(t)\, dt = f(s), \qquad |s| < 1, \tag{28}$$

provided that the kernel $K(t,s)$ is of the form

$$K(t,s) = \frac{L(t,s)}{t \pm s}. \tag{29}$$

Using quadratures similar to (17), we can reduce (28) to the following algebraic system,

$$(\mathbf{A} + \lambda \mathbf{C})\mathbf{y} = \mathbf{f}, \quad (A)_{j,i} = \frac{w_i}{t_i - s_j}, \quad (C)_{j,i} = \pi w_i K(t_i, s_j), \quad j = 1, \ldots, m,\ i = 1, \ldots, n, \tag{30}$$

where w_i are the quadrature weights, e.g. if $w(t) = (1-t^2)^{-1/2}$ then $w_i = 1/n$. The last algebraic system can be solved via iterative methods such as generalized conjugate gradient (Concus and Golub [3], Trummer [22]), Nyström's iterative variants (Gerasoulis [10]), and successive approximation (Tsamasphyros and Theocaris [23], Ioakimidis [16]). These iterative methods repeatedly compute products of the form $\mathbf{B_1 y}$, and the use of FAST will greatly improve the computational speed. Examples of equations with kernels similar to (29) can be found in Comninou [2], Tsamasphyros and Theocaris [23], Kaya [17], Elliott [5].

5. Approximate Algorithms. In this section we discuss two approximate algorithms which could be very useful for certain point distributions. One of these algorithms uses FAST while the other uses a power series approximation.

The procedure described in Section 2 only proves the existence of an algorithm whose computational complexity is smaller than that of the conventional algorithm. It does not show if FAST is practical, except in the case of the Chebyshev points for which FAST reduces to computing FFTs. How practical FAST is in general depends on how the points are distributed in the complex plane. For example, for the equidistant points $t_i = -1 + ih$, $i = 0, \ldots, n+1$, $h = 2/(n+1)$, interpolation is ill-conditioned for certain functional values (Dahlquist and Björck [4]). This implies that FAST could be unstable even for $n = 10$, since in step 5 of the algorithm interpolation is used. Moreover, even if the algorithm is stable, the constants for the time complexity could be larger than 10. Therefore, FAST could become faster than direct multiplication only if $n \geq 1000$. In such cases, an alternative implementation, or even an approximate algorithm, might be a better choice. In the case of the equidistant points one does not have to use FAST at all, since $t_i - t_j = (i-j)h$ and therefore the sums in (15) can be computed via FFT convolutions directly in $O_A(n \log n)$ time. However, as we will see below, the usefulness of FAST is not limited to the Chebyshev points. As a matter of fact, it could be used with an

approximate algorithm to compute the products $\mathbf{Ty}$ or $\mathbf{B}_p\mathbf{y}$ to within an accuracy ε in $O_A(n\log n)$ time.

Let us consider the matrix $\mathbf{T}$ and assume that the points c_i lie on a sufficiently smooth closed curve in the complex plane. Points that lie on smooth curves arise in the approximation of Cauchy singular integral equations in the complex plane. Under this assumption, Reichel [19] has shown that $\mathbf{T}$ can be approximated by

$$\mathbf{T} \approx (\mathbf{B}^{(2)} - \mathbf{B}^{(0)} - \mathbf{C}_n)\mathbf{D}_n^{-1}, \tag{31}$$

where $\mathbf{D}_n$ and $\mathbf{B}^{(0)}$ are diagonal matrices, $\mathbf{B}^{(2)}$ is a low-rank matrix and $\mathbf{C}_n$ is a matrix of the same form as $\mathbf{T}$ having elements c_k equal to the nth roots of unity. The product $\mathbf{B}^{(2)}\mathbf{y}$ can be computed directly in $O_A(k^2)$ time, where k depends on the tolerance ε and on the smoothness of the closed curve and is usually much smaller than n. Reichel uses properties of circulant matrices to compute $\mathbf{C}_n\mathbf{y}$ in $O_A(n\log n)$ time. We show in the example below that FAST also computes $\mathbf{C}_n\mathbf{y}$ in $O_A(n\log n)$ time and more importantly that its implementation is stable for large n.

Example 2. Consider the nth roots of unity

$$c_k = \exp\left(\frac{2\pi i(k-1)}{n}\right), \qquad i = \sqrt{-1},\ k = 1,\dots,n.$$

For these points we have that

1. $g(x) = \prod_{k=1}^{n}(x - c_k) = x^n - 1$.
2. $g'(x) = nx^{n-1}$ and $g''(x) = n(n-1)x^{n-2}$.
3. $g'(c_k) = n/c_k$ and $g''(c_k) = n(n-1)/c_k^2$, $k = 1,\dots,n$, in $O_A(n)$ time.
4. $h(c_k) = y_k g'(c_k)$, $k = 1,\dots,n$, in $O_A(n)$ time.
5. Set $h(x) = \sum_{m=0}^{n-1} a_m x^m$ so that $h(c_{k+1}) = \sum_{m=0}^{n-1} a_m c_{k+1}^m$, $k = 0,\dots,n-1$, implying

$$a_m = n^{-1}\sum_{k=0}^{n-1} h(c_{k+1})\exp\left(\frac{-2\pi ikm}{n}\right), \qquad m = 0,\dots,n-1, \tag{32}$$

which can be computed via FFT in $O_A(n\log n)$ time (Aho et al. [1]).

6. From the last step we have that $h'(x) = \sum_{m=0}^{n-1} m a_m x^{m-1}$ so that

$$h'(c_{k+1}) = \exp\left(\frac{-2\pi ik}{n}\right)\sum_{m=0}^{n-1} m a_m \exp\left(\frac{2\pi ikm}{n}\right), \qquad k = 0,\dots,n-1, \tag{33}$$

which again can be computed via FFT in $O_A(n\log n)$ time.

7. Equation (9) computes $x_k = (h'(c_k) - \frac{1}{2}y_k g''(c_k))/g'(c_k)$, $k = 1,\dots,n$, in $O_A(n)$ time.

We can easily see that the above implementation of FAST requires $O_A(n\log n)$ time. The computational performance is similar to that of the Chebyshev points shown in Table 4.1. □

Finally, we describe a technique based on a power series expansion. This technique is similar to the one used by Greengard and Rokhlin [13] in particle simulation and Odlyzko and Schönhage [18] in the fast computation of the zeta function. We present the method in the case of $p = 1$ in (4) and under the assumption that the points are *well separated.* Two sets of points are well separated if there exist

points t_0, s_0 and a positive number $r > 0$ such that $|t_i - t_0| < r$, $|s_j - s_0| < r$ and $|t_0 - s_0| > 3r$ (Greengard and Rokhlin [13]). Under these assumptions we see that $|s_j - s_0|/|x - s_0| < \frac{1}{2}$ for $x = t_i$. Therefore, using the geometric series expansion, we obtain

$$f_1(x) = \sum_{j=1}^{n} \frac{y_j}{(x - s_0)\{1 - (s_j - s_0)/(x - s_0)\}} = \sum_{k=0}^{\infty} b_k (x - s_0)^{-k-1},$$
$$b_k = \sum_{j=1}^{n} y_j (s_j - s_0)^k. \tag{34}$$

We can easily see that

$$f_1(x) \approx f_1^m(x) = \sum_{k=0}^{m} b_k (x - s_0)^{-k-1},$$
$$|f_1(x) - f_1^m(x)| \le A \left(\frac{1}{2}\right)^m, \quad A = \sum_{j=1}^{n} |y_j|, \quad x = t_i. \tag{35}$$

Let us assume that $f_1(x)$ is to be approximated to within a precision ε. Then m must be chosen so that $m \approx -\log_2(\varepsilon/A)$. The total cost of arithmetic operations required to compute $f_1^m(t_i)$, $i = 1, \ldots, n$, is $O_A(mn)$. If $m \ll n$, then the complexity reduces to $O_A(n)$. Therefore, for the well-separated points this algorithm will be asymptotically faster than FAST, provided that ε is not very small.

If the points are not well separated, then this technique can still be applied. The sum in (34) can be split into subsums of well-separated and not well-separated (nearby) points. The computations for the nearby points can be performed directly, while we can use (35) for the well-separated points (see Greengard and Rokhlin [13] for details). The complexity of the algorithm depends on the homogeneity of the distribution of the points. For nonhomogeneous distributions, n might have to be extremely large before the algorithm becomes faster than direct multiplication. The applications reported in Greengard and Rokhlin [13], and Odlyzko and Schönhage [18], however, show that this algorithm could also be very useful for certain point distributions.

Acknowledgments. I would like to thank Gene Golub for introducing us to Trummer's problem and providing us with reference [7] and the referee for several valuable comments.

Department of Computer Science
Rutgers University
New Brunswick, New Jersey 08903

1. A. AHO, J. E. HOPCROFT & J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass., 1974.

2. M. COMNINOU, "The interface crack," *J. Appl. Mech.*, Transactions of ASME, 1977, pp. 631–636.

3. P. CONCUS & G. H. GOLUB, *A Generalized Conjugate Gradient Method for Nonsymmetric Systems of Linear Equations*, Lecture Notes in Economics and Mathematical Systems, v. 134, Springer-Verlag, Berlin, 1976, pp. 56–65.

4. G. DAHLQUIST & Å. BJÖRCK, *Numerical Methods*, Prentice-Hall, Englewood Cliffs, N.J., 1974.

5. D. ELLIOTT, "The numerical treatment of singular integral equations–A review," *Treatment of Integral Equations by Numerical Methods* (C. T. H. Baker and G. F. Miller, eds.), Academic Press, New York, 1982, pp. 297–312.

6. F. ERDOGAN & G. D. GUPTA, "On the numerical solution of singular integral equations," *Quart. Appl. Math.*, v. 29, 1972, pp. 525–534.

7. N. GASTINEL, "Inversion d'une matrice généralisant la matrice de Hilbert," *Chiffres*, v. 3, 1960, pp. 149–152.

8. W. GAUTSCHI & J. WIMP, "Computing the Hilbert transform of a Jacobi weight function," *BIT*, v. 27, 1987, pp. 203–215.

9. A. GERASOULIS, "On the existence of approximate solutions for singular integral equations of Cauchy type discretized by Gauss-Chebyshev quadrature formulae," *BIT*, v. 21, 1981, pp. 377–380.

10. A. GERASOULIS, "Singular integral equations: Direct and iterative variant methods," *Numerical Solution of Singular Integral Equations* (Gerasoulis & Vichnevetsky, eds.), IMACS publication, 1984, pp. 133–141.

11. A. GERASOULIS, M. D. GRIGORIADIS & LIPING SUN, "A fast algorithm for Trummer's problem," *SIAM J. Sci. Statist. Comput.*, v. 8, 1987, pp. s135–s138.

12. G. H. GOLUB, "Trummer problem," *SIGACT News*, v. 17, 1985, No. 2, ACM Special Interest Group on Automata and Computability Theory, p. 17.2-12.

13. L. GREENGARD & V. ROKHLIN, *A Fast Algorithm for Particle Simulations*, Research report YALEU/DCS/RR-459, April 1986.

14. P. HENRICI, *Applied and Computational Complex Analysis*, III, Wiley, New York, 1986.

15. E. HOROWITZ, "A unified view of the complexity of evaluation and interpolation," *Acta Inform.*, v. 3, 1974, pp. 123–133.

16. N. I. IOAKIMIDIS, "Three iterative methods for the numerical determination of stress intensity factors," *Engrg. Fracture Mech.*, v. 14, 1981, pp. 557–564.

17. A. KAYA, *Applications of Integral Equations with Strong Singularities in Fracture Mechanics*, Ph.D. thesis, Lehigh University, 1984.

18. A. M. ODLYZKO & A. SCHÖNHAGE, *Fast Algorithms for Multiple Evaluations of the Riemann Zeta Function*, Technical Report, AT& T Bell Laboratories, Murray Hill, N.J., 1986.

19. L. REICHEL, *A Matrix Problem with Applications to Rapid Solution of Integral Equations*, Report, Department of Mathematics, University of Kentucky, Lexington, 1986.

20. V. ROKHLIN, "Rapid solution of integral equations of classical potential theory," *J. Comput. Phys.*, v. 60, 1985, pp. 187–207.

21. P. SWARZTRAUBER, FFTPACK, Netlib@anl-mcs, Private communication.

22. M. TRUMMER, "An efficient implementation of a conformal mapping method using the Szegö kernel," *SIAM J. Numer. Anal.*, v. 23, 1986, pp. 853–872.

23. G. TSAMASPHYROS & P. S. THEOCARIS, "A recurrence formula for the direct solution of singular integral equations," *Comput. Methods Appl. Mech. Engrg.*, v. 31, 1982, pp. 79–89.