SIAM J. SCI. STAT. COMPUT.
Vol. 13, No. 6, pp. 1418–1432, November 1992

A FAST ALGORITHM TO SOLVE NONHOMOGENEOUS CAUCHY–RIEMANN EQUATIONS IN THE COMPLEX PLANE*

PRABIR DARIPA†

Abstract. An algorithm is provided for the fast and accurate computation of the solution of nonhomogeneous Cauchy–Riemann equations in the complex plane in the interior of a unit disk. The algorithm is based on the representation of the solution in terms of a double integral, some recursive relations in Fourier space, and fast Fourier transforms. The numerical evaluation of the solution at N^2 points on a polar coordinate grid by straightforward summation for the double integral would require $O(N^2)$ floating point operations per point. Evaluation of these integrals has been optimized in this paper giving an asymptotic operation count of $O(\ln N)$ per point on the average. In actual implementation, the algorithm has even better computational complexity, approximately of the order of $O(1)$ per point. The algorithm has the added advantage of working in place, meaning that no additional memory storage is required beyond that of the initial data. The performance of the algorithm has been demonstrated on several prototype problems. The algorithm has applications in many areas, particularly fluid mechanics, solid mechanics, and quasi-conformal mappings.

Key words. complex variable, nonhomogeneous Cauchy–Riemann equation, Beltrami equation

AMS(MOS) subject classifications. 65E05, 30C99, 65D30

1. Introduction.

Cauchy–Riemann (CR) equations arise in many areas including fluid mechanics, solid mechanics, and electrostatics. These equations are given by

$$\phi_x = \psi_y, \qquad \phi_y = -\psi_x, \tag{1.1}$$

where real variables ϕ and ψ are functions of x and y. The CR equations suggest that the complex function $u = \phi + i\psi$ is an analytic function of the complex variable $\sigma = x + iy$:

$$u = g(\sigma). \tag{1.2}$$

This equation can be equivalently written as $u_{\bar{\sigma}} = 0$. Henceforth we refer to this equation as the CR equation in the complex plane. The CR equation plays an important role for two main reasons: the first is the fact that these equations model many problems in many areas of applications; the second is that the power of complex variable theory may be used in solving these problems. It should be noted that the CR equation is linear and homogeneous. Thus it is not surprising that CR equations arise due to some linear approximations or some nonlinear transformations of the original equations that actually arise in applied fields.

An obvious extension of the CR equation would be to add a nonhomogeneous term to it:

$$u_{\bar{\sigma}} = h(u, \sigma, \bar{\sigma}), \tag{1.3}$$

where the complex function h may be a nonlinear function of u and depends explicitly on σ and $\bar{\sigma}$. This equation naturally reduces to a CR equation when h is identically zero. Note that this equation is nonlinear as well as nonhomogeneous. A well-known special case of (1.3) is the following Beltrami equation (see [1] and [2]):

$$u_{\bar{\sigma}} = \mu(u) u_{\sigma}, \tag{1.4}$$

* Received by the editors November 21, 1990; accepted for publication (in revised form) September 8, 1991. This research was partially supported by National Science Foundation grant DMS-8803669.

† Department of Mathematics, Texas A & M University, College Station, Texas 77843.

where μ may depend on u, σ, and $\bar{\sigma}$. This equation arises in many instances, most notably in the transformation of a quasi-linear partial differential equation to canonical form [2], in quasi-conformal mapping [7], and in compressible fluid flow [6]. In fact, recent work has been done by solving these equations in the real as well as complex plane (see [5] and [6]) in the context of compressible fluid flow and quasi-conformal mapping [7]. It may also arise in other applied areas.

In this paper we are interested in constructing a fast algorithm to solve an appropriate boundary value problem associated with the following linearized version of (1.3):

$$u_{\bar{\sigma}} = f(\sigma, \bar{\sigma}), \tag{1.5}$$

in the unit disk $|\sigma| \leq 1$. The nonanalytic function f is assumed to satisfy a Hölder condition with exponent α in a unit disk Ω: $|\sigma| \leq 1$. Henceforth we refer to this equation as the nonhomogeneous Cauchy–Riemann (NCR) equation. Below, we denote $f(\sigma, \bar{\sigma})$ by $f(\sigma)$.

In [6], Daripa introduced a numerical method to solve this problem. The method is based on splitting the solution of (1.5) as (see also [2])

$$u(\sigma) = u^p(\sigma) + u^a(\sigma), \tag{1.6}$$

where $u^a(\sigma)$ is the analytic part of the solution and $u^p(\sigma)$ is the nonanalytic part of the solution. The nonanalytic part $u^p(\sigma)$ is then given by

$$u^p(\sigma) = -\frac{1}{\pi} \int\int_{\Omega} \frac{f(\zeta)}{\zeta - \sigma} d\xi d\eta, \tag{1.7}$$

where $\zeta = \xi + i\eta$.

The analytic part admits a Taylor series representation which can be efficiently computed using the fast Fourier transform (FFT) (see [3], [4], [6], and [8]). However, there are two main difficulties in the computation of the nonanalytic part $u^p(\sigma)$. The first has to do with the singularity of the integrand in (1.7) at $\zeta = \sigma$. In [6, Appendix C], Daripa proposed a method of desingularizing the integral. This desingularized version of the integral can then be directly integrated, but with poor accuracy, as is explained in §5. The second has to do with the algorithmic complexity of straightforward integration. The straightforward computation of the double integral in (1.7) requires performing an integral with an operation count of the order of $O(N^2)$ for each node in the discretization. There are N^2 such nodes in the discretization of the domain and hence N^2 such integrals to be evaluated. Thus this method of evaluation is computationally very intensive. These are the main drawbacks of the method. However, our goal in this paper is to develop an efficient and accurate algorithm for evaluating this integral and to thus make this a desirable alternative to the above approach.

The process of evaluating these integrals is optimized in this paper, giving a net operation count of the order of $O(N^2 \ln N)$ for N^2 points. This algorithm has the added advantage of working in place, meaning that no additional memory storage is required beyond that of the initial data.

Our method is basically a recursive routine in Fourier space that divides the entire domain (the interior of the unit disk) into a collection of annular regions and expands the integral in Fourier series in angular direction with radius-dependent Fourier coefficients. A set of exact recursive relations are derived which are then used to produce the Fourier coefficients of the integral. These recursive relations involve appropriate scaling of one-dimensional integrals in annular regions. The desired integrals at all N^2 grid points are then easily obtained from the Fourier coefficients by the FFT.

The algorithm developed in this paper also provides a constructive analytical method for evaluating certain integrals that are otherwise difficult to evaluate. An example is provided in Appendix B.

The rest of the paper is structured as follows. Section 2 develops the mathematical foundation of the efficient algorithm to evaluate the particular solution $u^p(\sigma)$ within the unit disk. The fast algorithm and the algorithmic complexity are discussed in §3. In §4, an area integral is evaluated corroborating the algorithmic steps. Section 5 discusses the computational results in evaluating the particular solution using our algorithm of §4. In §6, Dirichlet problems associated with the NCR equation (1.5) and computations of their solution are discussed. We conclude in §7.

In a sequel, we shall apply our algorithm to equations of compressible fluid flow.

2. Mathematical foundation of the algorithm.

In this section, we develop the theory needed to construct an efficient algorithm for evaluating the double integral that appears in (1.7).

In the following, we use the notations Ω_r: $|\sigma| \leq r < 1$, $\Omega_{\bar{r}}$: $\Omega \backslash \Omega_r$, and Ω_{ij}: $r_i \leq |\sigma| \leq r_j$. The following theorem is crucial for the later development of the algorithm.

THEOREM 2.1. *The particular solution of $u_{\bar{\sigma}} = f(\sigma)$ with $\sigma = re^{i\alpha}$ can be written as*

$$u^p(\sigma) = \sum_{n=-\infty}^{\infty} c_n(r) e^{in\alpha}, \tag{2.1}$$

where

$$c_n(r) = \begin{cases} \dfrac{1}{\pi} \displaystyle\iint_{\Omega_r} f(\zeta) \left(\frac{r}{\zeta}\right)^n \left(\frac{1}{\zeta}\right) d\xi d\eta, & n < 0, \\[2ex] -\dfrac{1}{\pi} \displaystyle\iint_{\Omega_{\bar{r}}} f(\zeta) \left(\frac{r}{\zeta}\right)^n \left(\frac{1}{\zeta}\right) d\xi d\eta, & n \geq 0. \end{cases} \tag{2.2}$$

Proof. If we introduce the notation

$$P_n(r,\zeta) = \int_0^{2\pi} \frac{e^{-in\alpha}}{\zeta - re^{i\alpha}} d\alpha, \tag{2.3}$$

then it follows from (1.7) and (2.1) that

$$-\pi c_n(r) = \frac{1}{2\pi} \iint_{\Omega} f(\zeta) P_n(r,\zeta) d\xi d\eta. \tag{2.4}$$

The integral in (2.3) can be evaluated using complex variables. It is an elementary exercise in complex variable theory to show that

$$P_n(r,\zeta) = -2\pi r^n S_n(\zeta), \tag{2.5}$$

where

$$S_n(\zeta) = -\delta(n)\zeta^{-(n+1)} + \begin{cases} \zeta^{-(n+1)}, & |\zeta| < |\sigma|, \\ 0.5\zeta^{-(n+1)}, & |\zeta| = |\sigma|, \\ 0, & |\zeta| > |\sigma|. \end{cases} \tag{2.6}$$

In (2.6), $\delta(n) = 0$ for $n < 0$ and $\delta(n) = 1$ for $n \geq 0$. Substitution of (2.5) into (2.4) yields the desired result, i.e., (2.2).

Remark 2.1. The above theorem can also be derived by first expanding $\frac{1}{\zeta-\sigma}$ in power of $\frac{\zeta}{\sigma}$ and $\frac{\sigma}{\zeta}$, followed by integration in the complex plane.

COROLLARY 2.1. *Suppose that $\zeta = \rho e^{i\theta}$ and $f(\zeta)$ has Fourier coefficients $f_n(\rho)$; then it easily follows that the coefficients $c_n(r)$ in (2.2) can be written as*

$$c_n(r) = \begin{cases} 2\int_0^r f_{n+1}(\rho)\left(\frac{r}{\rho}\right)^n d\rho, & n<0, \\ -2\int_r^1 f_{n+1}(\rho)\left(\frac{r}{\rho}\right)^n d\rho, & n\geq 0. \end{cases} \tag{2.7}$$

COROLLARY 2.2. *It follows directly from (2.7) that $c_n(1) = 0$ for $n \geq 0, c_n(0) = 0$ for all $n \neq 0$. It also follows from the Fourier expansion of $f(\zeta)$ that $f_n(0) = 0$ for $n \neq 0$, and $f_0(0) = f(0)$.*

COROLLARY 2.3. *Let $r_j > r_i$. Define*

$$c_n^{ij} = 2\int_{r_i}^{r_j} f_{n+1}(\rho)\left(\frac{R}{\rho}\right)^n d\rho, \tag{2.8}$$

where

$$R = \begin{cases} r_i, & n\geq 0, \\ r_j, & n<0. \end{cases} \tag{2.9}$$

After some algebraic manipulation it follows from (2.7) that

$$c_n(r_j) = \left(\frac{r_j}{r_i}\right)^n c_n(r_i) + c_n^{ji}, \qquad n<0, \tag{2.10}$$

and

$$c_n(r_i) = \left(\frac{r_i}{r_j}\right)^n c_n(r_j) - c_n^{ij}, \qquad n\geq 0. \tag{2.11}$$

COROLLARY 2.4. *Let $0 = r_1 < r_2 < r_3 \cdots < r_M = 1$. It follows from recursive applications of (2.10) and (2.11) and from using Corollary 2.2 that*

$$c_n(r_l) = \begin{cases} \sum_{i=2}^{l}\left(\frac{r_l}{r_i}\right)^n c_n^{i-1,i}, & \text{for } n<0 \text{ and } l = 2,\cdots,M, \\ -\sum_{i=l}^{M-1}\left(\frac{r_l}{r_i}\right)^n c_n^{i,i+1}, & \text{for } n\geq 0 \text{ and } l = 1,\cdots,M-1. \end{cases} \tag{2.12}$$

3. The fast algorithm.

We construct the fast algorithm based on the theory of §2. The unit disk is discretized using $M \times N$ lattice points with M equidistant points in the radial direction and N equidistant points in the circular direction.

Initialization. Choose M and N. Define $K = \frac{N}{2}$.
Step 1. For $l \in [1, M]$ and $n \in [-K+1, K]$, compute the Fourier coefficients $f_n(r_l)$ of $f(\zeta)$ from the known values of $f(\zeta = r_j e^{2\pi ik/N}), j = 1, \cdots, M; k = 1, \cdots, N$.

Step 2. Compute $c_n^{i,i+1}, i \in [1, M-1]$ for $n \in [-K, K-1]$ using (2.8).

Step 3. Compute the Fourier coefficients $c_n(r_l), n \in [-K, K-1], l \in [1, M]$ using the relations (2.10) and (2.11).

```
set $c_n(r_M) = 0 \ \forall \ n \in [0, K-1]$
do $n = 0, \cdots, K-1$
    do $l = M-1, \cdots, 1$
    Use (2.11) of Corollary 2.4 to compute $c_n(r_l)$.
    $c_n(r_l) = \left(\frac{r_l}{r_{l+1}}\right)^n c_n(r_{l+1}) - c_n^{l,l+1}$
    enddo
enddo

set $c_n(r_1) = 0 \ \forall \ n \in [-K, -1]$
do $n = -K, \cdots, -1$
    do $l = 2, \cdots, M$
    Use (2.10) of Corollary 2.4 to compute $c_n(r_l)$
    $c_n(r_l) = \left(\frac{r_l}{r_{l-1}}\right)^n c_n(r_{l-1}) + c_n^{l-1,l}$
    enddo
enddo
```

Step 4. Finally compute $u^p(\sigma = r_j e^{2\pi ik/N}), j \in [1, M], k \in [1, N]$ using a truncated version of (2.1).

3.1. The algorithmic complexity.

Here we consider the computational complexity of the above algorithm. We discuss the asymptotic operation count, the asymptotic time complexity, and asymptotic storage requirement, in that order. In Steps 1 and 4 above, there are 2M FFTs of length N and all other computations in Steps 2 and 3 are of lower order. With each FFT of length N contributing $N \ln N$ operations, the asymptotic operation count and hence the asymptotic time complexity is $O(MN \ln N)$.

The algorithm requires storing the MN Fourier coefficients $f_n(r_l)$ in Step 1, the MN Fourier coefficients $c_n(r_l)$ in Step 3, and the MN values of the desired u^p at MN grid points in Step 4. Therefore the asymptotic storage requirement is $O(MN)$.

Remark 3.1. The computation c_n^{ij} in Step 2 can be embedded within the inner do-loops of Step 3, thus avoiding the storage requirement for these. Note that we present the algorithm in the form shown above for the sake of clarity and without any sacrifice in the asymptotic time complexity.

4. Integral evaluation using the algorithmic steps.

We illustrate the algorithmic steps mentioned above through an explicit example. There are three main reasons for doing this. The first is to give an explicit exposition of the operations in each of the algorithmic steps in detail, the second is to show the power of the algorithm to evaluate the integral analytically when function f is known explicitly in terms of the complex coordinates, and the third is to generate an explicit example to test the accuracy and the algorithmic complexity at each step of the algorithm by explicit numerical calculations. This is also useful for debugging various stages of the computation.

We first summarize our result in the following note, proof of which follows.

Note. If

$$f(\sigma) = \sigma^p \bar{\sigma}^q \tag{4.1}$$

where p and q are constants, so that $k = p - q$ is an integer, then the integral $u^p(\sigma)$ in (1.7) is given by the following.

For $k = p - q \geq 1$,

$$u^p(\sigma) = \begin{cases} \dfrac{\sigma^{(k-1)}}{q+1}(|\sigma|^{2(q+1)} - 1) & \text{if } q \neq -1, \\ 2\sigma^{k-1} \ln |\sigma| & \text{if } q = -1, \end{cases} \tag{4.2}$$

and, **For** $k = p - q < 1$,

$$u^p(\sigma) = \begin{cases} \dfrac{\sigma^{k-1}}{q+1}|\sigma|^{2(q+1)} & \text{if } q > -1, \\ \infty & \text{if } q \leq -1. \end{cases} \tag{4.3}$$

Proof. We prove this through the use of the algorithm described in §3.

Step 1. Since $\zeta = \rho e^{i\theta}$, we have from (4.1),

$$f_n = \begin{cases} \rho^{p+q} & \text{if } n = k, \\ 0 & \text{if } n \neq k. \end{cases} \tag{4.4}$$

Step 2. From (2.8) and (4.4)

$$c_n^{ij} = 0 \quad \text{for} \quad n \neq k - 1. \tag{4.5}$$

The nonzero coefficient c_{k-1}^{ij} is evaluated as follows according to the algorithm.

$$c_{k-1}^{i,i+1} = 2R^{k-1} \int_{r_i}^{r_{i+1}} \rho^{p+q+k-1} d\rho. \tag{4.6}$$

Since $k = p - q$, upon integration, (4.6) becomes

$$c_{k-1}^{i,i+1} = 2R^{k-1} \begin{cases} \dfrac{1}{2(q+1)} \left(r_{i+1}^{2(q+1)} - r_i^{2(q+1)}\right) & \text{if } q \neq -1, \\ \ln(r_{i+1}/r_i) & \text{if } q = -1. \end{cases} \tag{4.7}$$

Step 3. It follows from (2.10), (2.11), (4.5), and Corollary 2.2 that

$$c_n(r_j) = 0 \quad \text{for } n \neq k - 1, \quad \text{and} \quad j = 1, 2, \cdots, M. \tag{4.8}$$

The calculation of the coefficient $c_{k-1}(r_j), j = 1, 2, \cdots, M$, according to our algorithm, follows. There are two separate cases to be considered depending on whether $q = -1$ or $q \neq -1$.

Case 1. $q = -1$.

(i) **For** $k < 1$, upon using (4.7) in (2.12) we obtain, after some manipulation,

$$c_{k-1}(r_l) = 2r_\ell^{k-1} \sum_{i=2}^{\ell} \ln\left(\frac{r_i}{r_{i-1}}\right) = \infty, \tag{4.9}$$

where $0 \leq r_\ell \leq 1$. To arrive at the last equality in (4.9) we have used

$$\sum_{i=2}^{\ell} \ln\left(\frac{r_i}{r_{i-1}}\right) = \sum_{i=2}^{\ell} (\ln r_i - \ln r_{i-1}) = \ln r_\ell - \ln 0 = \infty. \tag{4.10}$$

(ii) **For** $k \geq 1$, upon using (4.7) in (2.12) we obtain, after some manipulation,

$$\begin{aligned} c_{k-1}(r_\ell) &= -2r_\ell^{k-1} \sum_{i=\ell}^{m-1} \ln\left(\frac{r_{i+1}}{r_i}\right) \\ &= 2r_\ell^{k-1} \ln(r_\ell), \end{aligned} \tag{4.11}$$

where $0 \leq r_\ell \leq 1$. To arrive at the last equality in (4.11) we have used

$$\begin{aligned} \sum_{i=\ell}^{m-1} \ln\left(\frac{r_{i+1}}{r_i}\right) &= (\ln(r_m) - \ln(r_{m-1})) + (\ln(r_{m-1}) - \ln(r_{m-2})) \\ &\quad + \cdots + (\ln r_{\ell+1} - \ln r_\ell) \\ &= \ln 1 - \ln r_\ell = -\ln r_\ell. \end{aligned} \tag{4.12}$$

Case 2. $q \neq -1$.

(i) **For** $k < 1$, upon using (4.7) in (2.12) we obtain, after some manipulation,

$$\begin{aligned} c_{k-1}(r_\ell) &= \frac{r_\ell^{k-1}}{q+1} \sum_{i=2}^{\ell} (r_i^{2(q+1)} - r_{i-1}^{2(q+1)}) \\ &= \frac{r_\ell^{k-1}}{q+1} (r_\ell^{2(q+1)} - r_1^{2(q+1)}) \\ &= \begin{cases} \frac{r_\ell^{k-1}}{q+1} r_\ell^{2(q+1)} & \text{if } q > -1, \\ \infty & \text{if } q < -1, \end{cases} \end{aligned} \tag{4.13}$$

where we have used

$$\sum_{i=2}^{\ell} (r_i^{2(q+1)} - r_{i-1}^{2(q+1)}) = r_\ell^{2(q+1)} - r_1^{2(q+1)}. \tag{4.14}$$

(ii) **For** $k \geq 1$, upon using (4.7) in (2.12) we obtain after some manipulation

$$\begin{aligned} c_{k-1}(r_\ell) &= -\frac{r_\ell^{k-1}}{q+1} \sum_{i=m-1}^{\ell} (r_{i+1}^{2(q+1)} - r_i^{2(q+1)}) \\ &= \frac{r_\ell^{k-1}}{q+1} (r_\ell^{2(q+1)} - 1), \end{aligned} \tag{4.15}$$

where $0 \le r_\ell \le 1$. To obtain the last equality in (4.15) we have used

$$\sum_{i=\ell}^{m-1} (r_{i+1}^{2(q+1)} - r_i^{2(q+1)}) = r_m^{2(q+1)} - r_\ell^{2(q+1)} = 1 - r_\ell^{2(q+1)}. \tag{4.16}$$

We can summarize the calculations of Step 3 by rewriting (4.9) and (4.13) into

(i) **For** $k < 1$,

$$c_{k-1}(r_\ell) = \begin{cases} \dfrac{r_l^{k-1}}{q+1} r_l^{2(q+1)} & \text{if } q > -1, \\ \infty & \text{if } q \le -1, \end{cases} \tag{4.17}$$

and by rewriting (4.11) and (4.15) into

(ii) **For** $k \ge 1$

$$c_{k-1}(r_\ell) = \begin{cases} \dfrac{r_\ell^{k-1}}{q+1}(r_\ell^{2(q+1)} - 1), & q \ne -1, \\ 2r_\ell^{k-1}\ln(r_\ell), & q = -1. \end{cases} \tag{4.18}$$

Step 4. Using the Fourier coefficients given by (4.8), (4.9), (4.11), (4.13), and (4.15) in the Fourier series (2.1), we obtain our results (4.2) and (4.3).

Remark 4.1. The above method of integral evaluation is meant only to show the algorithmic steps explicitly. There are easier *analytical* methods to evaluate the above integral. For example, it is much easier to evaluate the coefficient c_{k-1} directly from (2.7).

Remark 4.2. We note that $u^p(\sigma)$ in the above example can be determined directly to within an arbitrary additive analytic function by simply integrating (4.1) with respect to the conjugate of the complex coordinate.

Remark 4.3. The above calculation is exact. In actual implementation, the errors will arise from finite truncation of Fourier series and approximate evaluation of the one-dimensional integrals. However, in this particular example there is only one mode, namely, $(p-q)$th mode, in $f(\sigma)$ (see (4.1)). Therefore, the error due to discrete Fourier approximation in this example is zero, provided this mode is included in the truncated Fourier series, i.e., $N > p$. In actual implementation, the errors due to numerical evaluation of the integrals depend on the method of integration and the values of p and q. In our implementation, we use trapezoidal rule and thus the error in Step 3 of the algorithm can be made zero if q is chosen to be zero in (4.1) (regardless of how many points are chosen in the radial integration).

The following two remarks have to do with the divergence of the double integral (1.7) for some specific choices for $f(\sigma)$. For the specific functional form (4.1) for $f(\sigma)$, the divergence of the double integral depends on the values of p and q.

Remark 4.4. From (4.3), it follows that $u^p(\sigma)$ blows up at all values of σ within the unit disk if $p - q - 1 < 0$ and $q \le -1$, or equivalently, if $p < q + 1 \le 0$.

Remark 4.5. From (4.2), it follows that $u^p(r = 0)$ blows up if $p - q - 1 \ge 0$ and $q = -1$, or equivalently, if $p \ge q + 1 = 0$.

5. Numerical results I.

A computer program has been written which can compute the integral using the fast algorithm of this paper or using the naive method of directly integrating the double integral in [6, Appendix C, eq. (C.4)]. The program has been tested with various functions $f(\sigma)$ in (1.7). However, we present the performance of our fast algorithm using the example of §4. Since the exact value of the integral is available in this case, the relative maximum error in the numerical computation can easily be calculated.

Computations were carried out for different radial and angular grid spacings on a Cray Y-MP at Texas A&M University in single precision. The computations were performed in two ways: (i) using the fast algorithm, and (ii) using the direct method. We compare these two methods by monitoring CPU time and relative maximum error for various values of M and N. We summarize our numerical results below.

Computations were performed for various values of p and q such that $(p-q)$ is an integer. The results obtained with $q=0$ and with $N>p$ using the fast algorithm were accurate up to seven decimal places in single precision regardless of the number of radial grid points. In this calculation, the effect of the number of radial grid points is zero, as it should be according to Remark 4.3. The effect of the number of angular grid points is also zero since this number N has been chosen to be greater than p. Thus, the only error in this case is due to truncation error. However, there were no signs of truncation errors within seven decimal places in this case.

The results of the computations using $p=3$ and $q=1$ in the choice $f(\delta)$ is summarized in Table 1. The number of angular grid points N is kept constant at 17. The first column contains the number of radial grid points. The second and the third columns contain the CPU times T_{fast}, required by the fast algorithm of the present paper, and T_{dir}, required by the direct method, respectively. The fourth and fifth columns contain the maximum relative errors δ_{fast} and δ_{dir} in these two methods, respectively.

TABLE 1
CPU *times in seconds and maximum relative errors on Cray* Y-MP. *The terms within parentheses are approximate estimates.*

M	T_{fast}	T_{dir}	δ_{fast}	δ_{dir}
51	0.00913	1.61807	2.878941E-04	7.7745298E-02
101	0.01662	6.40785	7.1945221E-05	8.438147E-02
151	0.02408	14.3590	3.197717E-05	8.666666E-02
201	0.03157	25.41848	1.7987853E-05	8.78808483E-02
251	0.03987	39.65647	1.1513032E-05	8.88519315E-02
501	0.07786	158.16779	2.8797382E-06	8.9932327E-02
1001	0.15409	(632.67116)	7.2160367E-07	–

Remark 5.1. The fast algorithm of the present paper takes only 0.154 seconds of CPU time when $M=1001$. The CPU time when using the direct method with $M=1001$ is estimated by extrapolation (shown within parentheses in Table 1). It was not considered practical to use approximately 632 seconds of Cray CPU time to produce an exact value of this CPU time.

The following observations can be made about Table 1.

(i) The CPU time required by the fast algorithm increases linearly with M. In contrast, the CPU time required by the direct method increases quadratically with M.

(ii) The relative maximum error δ_{fast} decreases with increasing M. This is because the error in the numerical integration by the trapezoidal method in Step 2 decreases with increasing M.

(iii) The relative maximum error δ_{dir} decreases very slowly with increasing M. In this case, the function $f(\zeta)$ is very poorly resolved by only 17 points in the angular direction. Most of the error is probably due to this poor resolution.

(iv) The accuracy of the fast algorithm is remarkable. In contrast, the direct method has very poor accuracy.

The results of similar computations using $M = 101$ are shown in Table 2 for varying values of N. The first column contains the number of angular grid points N.

TABLE 2
CPU *times in seconds and maximum relative errors on Cray* Y-MP. *The terms within parentheses are approximate estimates.*

N	T_{fast}	T_{dir}	δ_{fast}	δ_{dir}
17	0.01662	6.40785	7.1945221E-05	8.438147E-02
33	0.03216	24.32479	7.1974139E-05	2.592902E-02
65	0.06680	93.81604	7.1972282E-05	6.968137E-03
129	0.13908	376.73983	7.1977357E-05	1.576032E-03
257	0.30031	(1500.00)	7.1972305E-05	6.520931E-04
513	0.64261	(6000.00)	7.1977943E-05	4.129121E-04
1025	1.36825	(24000.00)	7.1972308E-05	2.152631E-04

Remark 5.2. The Cray Y-MP CPU seconds shown within parentheses are approximate and were estimated from computations on the local MIPS computer. The corresponding errors were obtained on the local MIPS computer. The errors on Cray Y-MP and on the local MIPS computer for the same problem agree up to five decimal places.

The following observations can be made about Table 2.

(i) The CPU time required by the fast algorithm increases superlinearly with N and is less than the theoretical asymptotic estimate of $N \ln N$. In contrast, the CPU time required by the direct method increases quadratically with N.

(ii) The relative maximum error δ_{fast} does not change with changing N. This is expected since the values of N used are greater than $(p-q) = 2$ (see Remark 4.3).

(iii) The relative maximum error δ_{dir} decreases with increasing N. However, the accuracy of the fast algorithm is much better.

6. Dirichlet problems for nonhomogeneous Cauchy–Riemann equations.

This section shows the application of our fast algorithm of §3 on a Dirichlet linear boundary value problem. Computations of nonlinear and other types of boundary value problems associated with (1.5) using our fast algorithm are under way and will be addressed elsewhere in detail. This section has been kept as brief as possible. We consider solving the following Dirichlet problem in the interior of a unit disk.

$$\text{(P)} \qquad \begin{cases} u_{\bar\sigma} = f(\sigma, \bar\sigma), & |\sigma| \le 1, \\ \text{Real } [u(\sigma = e^{i\alpha})] = u_0(\alpha), & 0 \le \alpha \le 2\pi, \\ \text{Imag } [u(\sigma = 0)] = v_0. \end{cases}$$

We are interested in finding the solution of (P) in the entire domain.

It follows from the solution (1.6) of (1.5) that the above problem is equivalent to solving the following problem:

$$\text{(RP)}\qquad \begin{cases} u^a_{\bar\sigma} = 0, & |\sigma| \le 1, \\ \text{Real }[u^a(\sigma = e^{i\alpha})] = u_0(\alpha) - u^p(\alpha), & 0 \le \alpha \le 2\pi, \\ \text{Imag }[u^a(\sigma = 0)] = v_0 - u^p(\sigma = 0). \end{cases}$$

Once this problem has been solved, the solution of the problem (P) is constructed from (1.6). Therefore our method of solving (P) involves the following three steps:

(1) First, find the particular solution $u^p(\sigma)$ in the entire domain using the algorithm of the previous section. As shown previously, the algorithmic complexity of this stage of the computation is at worst $MN \ln N$ with MN grid points in the discretization of the unit disk.

(2) Second, construct the analytic function $u^a(\sigma)$ by solving the reduced problem (RP).

(3) Third, construct the solution of the original problem (P) by adding the above two solutions according to (1.6).

Note that the algorithmic complexity in solving the reduced problem (RP) must not exceed $MN \ln N$ so that the computational complexity of the entire calculation remains the same. There are many efficient algorithms that can be used to solve this problem. We use the following simple procedure in constructing the solution of the reduced problem (RP) at MN lattice points.

(i) The solution is expressed in terms of Taylor series

$$u^a(\sigma) = \sum_{n=0}^{\infty} c_n \sigma^n. \tag{6.1}$$

(ii) The FFT is used to construct the complex conjugate of the specified boundary values and the Fourier coefficients of the Taylor series. This step takes $N \ln N$ operations with N points on the boundary of the unit disk.

(iii) Next, the Fourier coefficients are used in constructing the solution $u^a(\sigma)$ at all interior points by using the FFT. This step takes $MN \ln N$ operations with M divisions in the radial direction.

Note that this approach has an asymptotic operation count of the order $MN \ln N$. Thus overall complexity is retained at the desired level.

Remark 6.1. More efficient algorithms can be constructed to solve the reduced problem (RP). We will implement better algorithms for this part of the calculation in the future.

6.1. Numerical results II.

The following problems were numerically solved using the above steps.

Example 1.

$$\text{(E1)}\qquad \begin{cases} u_{\bar\sigma} = 1, & |\sigma| \le 1, \\ \text{Real }[u(\sigma = e^{i\alpha})] = 2\cos\alpha, & 0 \le \alpha \le 2\pi, \\ \text{Imag }[u(\sigma = 0)] = 0. \end{cases}$$

The exact solution to this problem is

$$u(\sigma) = \sigma + \bar\sigma, \qquad |\sigma| \le 1. \tag{6.2}$$

Example 2.

$$\text{(E2)} \qquad \begin{cases} u_{\bar\sigma} = -1, & |\sigma| \le 1, \\ \text{Real } [u(\sigma = e^{i\alpha})] = 0, & 0 \le \alpha \le 2\pi, \\ \text{Imag } [u(\sigma = 0)] = 0. \end{cases}$$

The exact solution to this problem is

$$u(\sigma) = \sigma - \bar\sigma, \qquad |\sigma| \le 1. \tag{6.3}$$

Example 3.

$$\text{(E3)} \qquad \begin{cases} u_{\bar\sigma} = \sigma^k, & |\sigma| \le 1, \\ \text{Real } [u(\sigma = e^{i\alpha})] = \cos\alpha, & 0 \le \alpha \le 2\pi, \\ \text{Imag } [u(\sigma = 0)] = 0, \end{cases}$$

where k is either 1 or 2. The exact solution to this problem is

$$u(\sigma) = \sigma + \bar\sigma\sigma^k - \sigma^{k-1}, \qquad |\sigma| \le 1. \tag{6.4}$$

The above three problems were numerically solved using our fast algorithm to obtain the solution at all $M \times N$ nodes with N=17 and M=101. The computed solutions were compared with the exact solutions for accuracy and the CPU times were recorded. These are shown in Table 3. The data for (E3) corresponds to $k = 2$ (see (E3) above). The first column contains the reference to one of the problems mentioned above. The second column contains the total CPU seconds (T_{tot}) required to solve these problems. The third and the fourth columns contain the breakup of T_{tot} into T_{h} (CPU seconds required to find $u^a(\sigma)$ at all interior points from the Taylor series (6.1)) and T_{nh} (CPU seconds required to find $u^p(\sigma)$ by evaluating the double integral using our algorithm of §3). The fifth column contains the maximum relative error δ.

TABLE 3
Computational results on boundary value problems (E1), (E2), *and* (E3) *using the fast algorithm.*

Examples	T_{tot}	T_{h}	T_{nh}	δ
E1	0.00932	0.00251	0.00681	5.6915101E-06
E2	0.00969	0.00269	0.00700	8.1740268E-06
E3	0.00932	0.00251	0.00681	7.3560102E-06

In Table 3, we note that our fast algorithm solves the boundary value problems within the entire unit disk with very good accuracy and within a fraction of a CPU second. The CPU seconds for computing the analytic part (T_{h}) is less than half of the CPU seconds T_{nh} needed to evaluate the particular solution $u^p(\sigma)$ at all interior points. Similar computations have been done on other types of boundary value problems with similar conclusions.

Remark 6.2. The computation of the analytic part can be accelarated by borrowing some of the ideas from [9], [12], and [13], which will be undertaken in the future.

7. Conclusions. The present work develops a fast algorithm to solve nonhomogeneous Cauchy–Riemann equations in the interior of a unit disk in the complex plane. It is based on computation of the particular solution from its Fourier coefficients. The recursive relations satisfied by these Fourier coefficients are derived, which is at the heart of the algorithm. The speedup provided by the algorithm is dramatic even for a moderate number of nodes in the domain. Our numerical experiments show that the actual CPU time requirements for the algorithm are even less than the asymptotic CPU estimate, which is very encouraging.

The algorithm has the limitation that the problem has to be solved within the interior of a unit disk. Therefore to construct a solution in a domain which is not circular, a conformal mapping of the domain to the interior of a unit disk must be done prior to the use of our algorithm. This should not be difficult, as there are many numerical methods these days to perform such conformal mapping (see [10] and [12]).

Prototype linear boundary problems have been solved here using the fast algorithm to provide some future directions. Nonlinear boundary value problems can be solved using this algorithm iteratively. Computations of nonlinear and other types of boundary value problems associated with the NCR equation using our fast algorithm are currently under way. The application of the algorithm is not limited to any particular field, thus its application to compressible fluid problems should be straightforward, since the compressible fluid flow equations already admit a formulation similar to (1.5) (see [6]). In fact, many compressible flow problems, including those solved in Woods [14] with the tangent gas approximation, can now be solved exactly, accurately, and very efficiently using the algorithm of this paper.

The algorithm presented here is suitable for implementation on a serial computer. However, the recursive set of equations of §2 has a structure suitable for implementation on a parallel computer, which will yield considerable savings in computational time depending on the number of processors. Construction of algorithms suitable for implementation on a parallel computer needs further work. Some of the ideas presented by Katzenelson [11] in connection with computational structure involving recursive relations may be useful here.

Appendix A. Application of the results of §2 in double integral evaluation. The results obtained in §2 provide an analytical technique to evaluate the double integral (1.8) with complicated $f(\zeta)$, which are otherwise difficult to evaluate analytically. We provide a prototype example.

Consider the following choice for the function $f(\zeta)$:

$$f(\zeta) = \zeta^p \sin\left(\frac{\zeta}{|\zeta|}\right) \quad \text{for} \quad p \geq 0. \tag{A.1}$$

Then the Fourier coefficients $f_n(r)$ are given by

$$f_n(r) = \frac{r^p}{2\pi} \int_0^{2\pi} e^{i(p-n)\theta} \sin(e^{i\theta}) d\theta. \tag{A.2}$$

Using $\sigma = e^{i\theta}$, (A.2) reduces to

$$f_n(r) = \frac{r^p}{2\pi i} \oint \sigma^{p-n-1} \sin \sigma d\sigma. \tag{A.3}$$

Using the residues, we have from (A.3),

$$f_n(r) = r^p E(n-p), \tag{A.4}$$

where

$$E(n-p) = \begin{cases} 0 & \text{if } (n-p) \le 0 \text{ or } (n-p) = \text{even integer}, \\ \dfrac{1}{(n-p)!} & \text{if } (n-p) = 1, 5, 9, \cdots, \\ \dfrac{-1}{(n-p)!} & \text{if } (n-p) = 3, 7, 11, \cdots. \end{cases} \tag{A.5}$$

Using (A.4) in (2.8) we obtain

$$c_{n-1}^{i,i+1} = \begin{cases} 0 & \text{for } n \le p, \\ \dfrac{2E(n-p)}{(n+p)} R^{n+1}(r_{i+1}^{n+p} - r_i^{n+p}) & \text{for } n > p. \end{cases} \tag{A.6}$$

It then follows from (2.12) that

$$c_{n-1}(r_j) = 0 \quad \text{for} \quad n \le p, \text{ and } \quad j = 1, 2, \cdots, M. \tag{A.7}$$

For $n > p$, upon using (A.6) in (2.12) we obtain after some manipulation,

$$c_{n-1}(r_l) = \frac{2E(n-p)}{n+p} r_\ell^{n-1}(r_\ell^{n+p} - 1). \tag{A.8}$$

Using (A.7), (A.8), and (A.5), we obtain

$$u^p(\sigma) = \sum_{n=p}^{\infty} \frac{2E(n+1-p)}{(n+p+1)!} r_\ell (r_\ell^{n+p+1} - 1) e^{in\theta}. \tag{A.9}$$

Using $n - p = m$ and (A.5), we can rewrite (A.9) as

$$\begin{aligned} u^p(\sigma) &= \sum_{m=0}^{\infty} \frac{2}{(4m+1)!(4m+2p+1)} \sigma^{4m+p}(|\sigma|^{4m+2p+1} - 1) \\ &\quad - \sum_{m=0}^{\infty} \frac{2}{(4m+3)!(4m+2p+3)} \sigma^{4m+p+2}(|\sigma|^{4m+2p+3} - 1). \end{aligned} \tag{A.10}$$

Appendix B. Desingularization of $\tau^p = -\frac{1}{\pi}\int\int_\Omega f(\zeta)/(\zeta - \sigma) d\xi d\eta$. The desingularized version of this integral appears in Appendix C [6]. However, the equations (C.8) and (C.9) of that paper were in error and should be corrected, respectively, as (B.2) and (B.3) of this appendix. We prescribe here a much easier procedure than the one in [6] to desingularize this integral.

The above integral can be desingularized in the following manner:

$$\begin{aligned} u^p &= -\frac{1}{\pi} \int\int_\Omega \frac{f(\zeta)}{\zeta - \sigma} d\xi d\eta \\ &= -\frac{1}{\pi} \int\int_\Omega \frac{f(\zeta) - f(\sigma)}{\zeta - \sigma} d\xi d\eta - \frac{f(\sigma)}{\pi} \int\int_\Omega \frac{d\xi d\eta}{\zeta - \sigma}. \end{aligned} \tag{B.1}$$

The second integral in (B.1) can be evaluated by using the integral evaluated in §4. With $p = 0$ and $q = 0$ in (4.1), we obtain the following from (4.3):

$$\int\int_\Omega \frac{d\xi d\eta}{\zeta - \sigma} = -\frac{\pi}{\sigma}|\sigma|^2. \tag{B.2}$$

From (B.1) and (B.2) we have

$$u^p = \bar{\sigma} f(\sigma) - \frac{1}{\pi} \int \int_\Omega \frac{f(\zeta) - f(\sigma)}{\zeta - \sigma} d\xi d\eta. \tag{B.3}$$

This desingularized version of the integral is suitable for numerical computation in the direct method (see also [6]).

Acknowledgments. The allocation of computer resources by the Texas A&M Supercomputer Center is gratefully acknowledged. It is a pleasure to thank Eric Grosse for making some useful suggestions that helped to reduce the length of the paper.

REFERENCES

[1] L. Bers, *Mathematical Aspects of Subsonic and Transonic Gas Dynamics*, John Wiley, New York, 1958.
[2] R. Courant and D. Hilbert, *Methods of Mathematical Physics, Vol.* II, John Wiley, New York, 1961.
[3] P. Daripa and L. Sirovich, *Exact and approximate gas dynamics using the tangent gas*, J. Comput. Phys., 62 (1986a), pp. 400–413.
[4] ———, *An inverse method for subcritical flows*, J. Comput. Phys., 63 (1986b), pp. 311–326.
[5] P. Daripa, *An exact inverse method for subcritical flows*, Quart. Appl. Math., XLVI (1988), pp. 505–526.
[6] ———, *On applications of a complex variable method in compressible flows*, J. Comput. Phys., 88(1990), pp. 337–361.
[7] ———, *On a numerical method for quasi-conformal grid generation*, J. Comput. Phys., 96(1991), pp. 296–307.
[8] P. R. Garabedian, *Supercritical Wing Sections* III, Springer-Verlag, New York, 1977.
[9] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulation*, J. Comput. Phys., 73 (1987), pp. 325–348.
[10] P. Henrici, *Applied and Computational Complex Analysis, Vol.* 3, John Wiley, New York, 1986.
[11] J. Katzenelson, *Computational structure of the N-body problem*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 787–815.
[12] S. T. O'Donnell and V. Rokhlin, *A fast algorithm for the numerical evaluation of conformal mapping*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 475–487.
[13] V. Rokhlin, *Rapid solution of integral equations of classical potential theory,* J. Comput. Phys., 60 (1985), pp. 187–207.
[14] L.C. Woods, *The Theory of Subsonic Plane Flow*, Cambridge University Press, New York, 1961.