ALGORITHM 651
Algorithm HFFT—High-Order Fast-Direct Solution of the Helmholtz Equation

RONALD F. BOISVERT
National Bureau of Standards

HFFT is a software package for solving the Helmholtz equation on bounded two- and three-dimensional rectangular domains with Dirichlet, Neumann, or periodic boundary conditions. The software is the result of combining new fourth-order accurate compact finite difference (HODIE) discretizations and a fast-direct solution technique (the Fourier method). In this paper we briefly describe the user interface to HFFT and present an example of its usage and several details of its implementation.

Categories and Subject Descriptors: G.1.8 [**Numerical Analysis**] Partial differential equations—*elliptic equations*; G.4 [**Mathematics of Computing**] Mathematical Software

General Terms: Algorithms

Additional Key Words and Pharses: Fast-direct method, finite differences, Fourier method, Helmholtz equation, high-order accuracy, HODIE method

1. INTRODUCTION

We describe a collection of Fortran programs, collectively called HFFT, which solve the Helmholtz equation

$$\Delta u + \lambda u = g$$

(λ constant) in two- and three-dimensional rectangular domains with any combination of Dirichlet (solution prescribed), Neumann (normal derivative prescribed), or periodic boundary conditions. The software computes fourth-order accurate solutions (for suitably smooth problems) using compact finite differencing techniques (the HODIE method); users can optionally request second-order accurate differences. The resulting system of equations is solved by the Fourier method.

A version of this software is also available in the ELLPACK system [15] as modules HODIE FFT and HODIE FFT 3D. Similar techniques are used by two other ELLPACK modules. In FFT 9-POINT [9], a fourth-order HODIE

Author's address: R. F. Boisvert, National Bureau of Standards, Scientific Computing Division, Tech A151, Gaithersburg, MD, 20899.
1987 ACM 0098-3500/87/0900-0235 $00.00

discretization with FACR(1) solution is used to solve the two-dimensional Dirichlet problem for the Helmholtz equation. In HODIE 27-POINT 3D [11, 10], a sixth-order HODIE discretization with a fast tensor-product solution technique is used to solve the three-dimensional Dirichlet problem for the Poisson equation ($\lambda = 0$). This software extends both of these, admitting more general boundary conditions and removing certain restrictions on grid sizes. Other available software for this problem include the FISHPAK subroutines SEPX4 and HW3CRT [1].

A complete description of the numerical methods employed in HFFT, along with computational comparisons with SEPX4 and HW3CRT are given in the companion paper [2]. Here we describe the user interface to HFFT and give certain details of its implementation.

2. DESCRIPTION OF THE SOFTWARE

2.1 User Interface

There are four user entry points into HFFT—HFFT2, HFFT2A, HFFT3, and HFFT3A. The routines HFFT2 and HFFT3 solve two and three-dimensional problems, respectively, presenting a user interface similar to that used by modules in the ELLPACK system. The software is based upon an equispaced grid defined on the rectangular domain (AX, BX) × (AY, BY) × (AZ, BZ). The grid points are (x_i, y_j, z_k), for $1 \le i \le$ NX, $1 \le j \le$ NY, $1 \le k \le$ NZ, where x_i = AX + $(i - 1)h$, y_j = AY + $(j - 1)h$, z_k = AZ + $(k - 1)h$, and h = (BX − AX)/(NX − 1) = (BY − AY)/(NY − 1) = (BZ − AZ)/(NZ − 1). The user interface for each case is summarized below; complete details are given in the initial comments for each subroutine.

CALL HFFT3 (COEFU, PRHS, BRHS, AX, BX, AY, BY, AZ, BZ, NX, NY, NZ, BCTY, IORDER, U, LDXU, LDYU, WORK, NWORK, INFO)

Input variables

COEFU	coefficient of u in the differential equation.
AX,BX	limiting values of x in domain (AX < BX).
AY,BY	limiting values of y in domain (AY < BY).
AZ,BZ	limiting values of z in domain (AZ < BZ).
NX,NY,NZ	number of grid lines in x, y, z, respectively (includes boundaries).
BCTY	integer array of length 6, indicating the type of boundary condition along x = BX, y = AY, x = AX, y = BY, z = BZ, z = AZ, in that order. Possible values are 1 for Dirichlet, 2 for Neumann, and 3 for periodic.
IORDER	order of accuracy of the discretization (2 or 4).
LDXU	first dimension of the array U exactly as declared in the calling program (must be at least NX + 2).

LDYU	second dimension of the array U exactly as declared in the calling program (must be at least NY + 2).
WORK	workspace array of size NWORK.
NWORK	length of the array WORK exactly as declared in the calling program (must be at least (NX + 1)(NY + 1) (NZ + 1)(IORDER − 2)/2 + 2(NX × NY + NX × NZ + NY × NZ) + 2(NX + NY + 1) + max(2NX × NY,2NX + 5NY + 4NZ + (NX + NZ)/2 + 29).

Output variables

U	array of size at least NX + 2 by NY + 2 by NZ + 2 containing the solution at grid points, i.e., U(i, j, k) = $u(x_i, y_j, z_k)$ for $1 \le i \le$ NX, $1 \le j \le$ NY, $1 \le k \le$ NZ. The extra rows and columns of U are used as workspace.
INFO	error flag. INFO = 0 indicates that the program ran to completion. INFO = $-k \neq 0$ ($1 \le k \le 14$) indicates that error condition k was detected, INFO = $k \neq 0$ ($1 \le k \le 2$) indicates warning k.

User-supplied functions

PRHS	function of (x, y, z) which returns the right side of the differential equation. Must be declared EXTERNAL in the calling program.
BRHS	function of (k, x, y, z) which evaluates the boundary condition at (x, y, z) on side k. The value returned depends upon BCTY(k). If BCTY(k) = 1, u is returned. If BCTY(k) = 2, u_x is returned for k = 1, 3, u_y is returned for k = 2, 4, and u_z is returned for k = 5, 6. BRHS will not be called with $k = m$ if BCTY(m) = 3. Must be declared EXTERNAL in the calling program.

CALL HFFT2 (COEFU, PRHS, BRHS, AX, BX, AY, BY, NX, NY, BCTY, IORDER, U, LDXU, WORK, NWORK, INFO)

Input variables

COEFU	coefficient of u in the differential equation.
AX,BX	limiting values of x in domain (AX < BX).
AY,BY	limiting values of y in domain (AY < BY).
NX,NY	number of grid lines in x, y, respectively (includes boundaries).
BCTY	integer array of length 4 indicating the type of boundary condition along x = BX, y = AY, x = AX, y = BY, in that order. Possible values are 1 for Dirichlet, 2 for Neumann, and 3 for periodic.
IORDER	order of accuracy of the discretization (2 or 4).
LDXU	first dimension of the array U exactly as declared in the calling program (must be at least NX + 2).

WORK	workspace array of size NWORK.
NWORK	length of the array WORK exactly as declared in the calling program (must be at least (NX + 1)(NY + 1) (IORDER − 2)/2 + 7(NX + NY) + NX/2 + 15).

Output variables

U	array of size at least NX + 2 by NY + 2 containing the solution at grid points, i.e., $U(i, j) = u(x_i, y_j)$ for $1 \le i \le$ NX, $1 \le j \le$ NY. The extra rows and columns of U are used as workspace.
INFO	error flag. INFO = 0 indicates that the program ran to completion. INFO = $-k \neq 0$ $(1 \le k \le 10)$ indicates that error condition k was detected, INFO = $k \neq 0$ $(1 \le k \le 2)$ indicates warning k.

User-supplied functions

PRHS	function of (x, y) which returns the right side of the differential equation. Must be declared EXTERNAL in the calling program.
BRHS	function of (k, x, y) which evaluates the boundary condition at (x, y) on side k. The value returned depends upon BCTY(k). If BCTY$(k) = 1$, u is returned. If BCTY$(k) = 2$, u_x is returned for $k = 1$, 3 and u_y is returned for $k = 2$, 4. BRHS will not be called with $k = m$ if BCTY$(m) = 3$. Must be declared EXTERNAL in the calling program.

The subroutines HFFT2 and HFFT3 set up calls to the lower level routines HFFT2A and HFFT3A, respectively. Users may call HFFT2A and HFFT3A directly; the subroutines differ in that users are required to prestore all required function values as is done in FISHPAK. The input provided to these routines is summarized below.

```
CALL HFFT3A    (COEFU, NX, NY, NZ, H, GH, LDXGH,
                LDYGH, BCTY, BD1, BD2, BD3, BD4, BD5, BD6,
                LDXBD, LDYBD, IORDER, U, LDYU, LDYU,
                WORK, NWORK, INFO)
```

Input variables

COEFU	coefficient of u in the differential equation.
NX,NY,NZ	number of grid lines in x, y, z (including boundaries).
H	distance between grid lines.
U	array of size NX + 2 by NY + 2 by NZ + 2 containing values of the function g (right side of the differential equation) at grid points, i.e., $U(i, j, k) = g(x_i, y_j, z_k)$, $1 \le i \le$ NX, $1 \le j \le$ NY, $1 \le k \le$ NZ. The extra rows and columns are used for working storage.

LDXU	first dimension of the array U, exactly as declared in the calling program.
GH	array of size NX + 1 by NY + 1 by NZ + 1 containing values of the function g at midpoints of the subsquares of the grid, i.e., GH$(i, j, k) = g(x_i + h/2, y_j + h/2, z_k + h/2)$ for $1 \le i \le$ NX $-$ 1, $1 \le j \le$ NY $-$ 1, $1 \le k \le$ NZ $-$ 1. The extra rows and columns are used for working storage. GH is not referenced when IORDER = 2.
BCTY	integer array of length 4 indicating the type of boundary condition along x = BX, y = AY, x = AX, y = BY, in that order. Possible values are 1 for Dirichlet, 2 for Neumann, and 3 for periodic.
BD1	array of size NY by NZ containing boundary condition values for x = BX, i.e., BD1$(j, k) = f(x_{\mathrm{NX}}, y_j, z_k)$ for $1 \le j \le$ NY, $1 \le k \le$ NZ. f is u or u_x depending on whether BCTY(1) is 1 or 2. BD1 is not referenced when BCTY(1) = 3.
BD2	array of size NX by NZ containing boundary condition values for y = AY, i.e., BD2$(i, k) = f(x_i, y_1, z_k)$ for $1 \le i \le$ NX, $1 \le k \le$ NZ. f is u or u_y depending on whether BCTY(2) is 1 or 2. BD2 is not referenced when BCTY(2) = 3.
BD3	array of size NY by NZ containing boundary condition values for x = AX, i.e., BD3$(j, k) = f(x_1, y_j, z_k)$ for $1 \le j \le$ NY, $1 \le k \le$ NZ. f is u or u_x depending on whether BCTY(3) is 1 or 2. BD3 is not referenced when BCTY(3) = 3.
BD4	array of size NX by NZ containing boundary condition values for y = BY, i.e., BD4$(i, k) = f(x_i, y_{\mathrm{NY}}, z_k)$ for $1 \le i \le$ NX, $1 \le k \le$ NZ. f is u or u_y depending on whether BCTY(4) is 1 or 2. BD2 is not referenced when BCTY(4) = 3.
BD5	array of size NX by NY containing boundary condition values for z = BZ, i.e., BD5$(i, j) = f(x_i, y_j, z_{\mathrm{NZ}})$ for $1 \le i \le$ NX, $1 \le j \le$ NY. f is u or u_z depending on whether BCTY(5) is 1 or 2. BD5 is not referenced when BCTY(5) = 3.
BD6	array of size NX by NY containing boundary condition values for z = AZ. i.e., BD6$(i, j) = f(x_i, y_j, z_1)$ for $1 \le i \le$ NX, $1 \le j \le$ NY. f is u or u_z depending on whether BCTY(6) is 1 or 2. BD6 is not referenced when BCTY(6) = 3.
LDXBD	first dimension of the arrays BD2, BD4, BD5, and BD6 exactly as declared in the calling program.
LDYBD	first dimension of the arrays BD1 and BD3 exactly as declared in the calling program.
IORDER	order of accuracy of the discretization (2 or 4).
WORK	workspace array of size NWORK.
NWORK	length of the array WORK exactly as declared in the calling program (must be at least (NX + 1)(NY + 1)(IORDER − 2) + (NX + 3)(NY + 5) + 5NY + (NX + NZ)/2 + 15.

Output variables

U	contains the solution at grid points, i.e., $U(i, j, k) = u(x_i, y_j, z_k)$ for $1 \le i \le NX$, $1 \le j \le NY$, $1 \le k \le NZ$.
INFO	error flag. INFO = 0 indicates that the program ran to completion. INFO = $-k \neq 0$ ($1 \le k \le 11$) indicates that error condition k was detected, INFO = $k \neq 0$ ($1 \le k \le 2$) indicates warning k.

CALL HFFT2A (COEFU, NX, NY, H, GH, LDXGH, BCTY, BD1, BD2, BD3, BD4, IORDER, U, LDXU, WORK, NWORK, INFO)

Input variables

COEFU	coefficient of u in the differential equation.
NX,NY	number of grid lines in x, y (including boundaries).
H	distance between grid lines.
U	array of size NX + 2 by NY + 2 containing values of the function g (right-hand side of the differential equation) at grid points, i.e., $U(i, j) = g(x_i, y_j)$. The extra rows and columns are used for working storage.
LDXU	first dimension of the array U, exactly as declared in the calling program.
GH	array of size NX + 1 by NY + 1 containing values of the function g at midpoints of the subsquares of the grid, i.e., $GH(i, j) = g(x_i + h/2, y_j + h/2)$ for $1 \le i \le NX - 1$, $1 \le j \le NY - 1$. The extra rows and columns are used for working storage. GH is not referenced when IORDER = 2.
BCTY	integer array of length 4 indicating the type of boundary condition along x = BX, y = AY, x = AX, y = BY, in that order. Possible values are 1 for Dirichlet, 2 for Neumann, and 3 for periodic.
BD1	array of size NY containing boundary condition values for x = BX, i.e., $BD1(j) = f(x_{NX}, y_j)$ for $1 \le j \le NY$. f is u or u_x depending on whether BCTY(1) is 1 or 2. BD1 is not referenced when BCTY(1) = 3.
BD2	array of size NX containing boundary condition values for y = AY, i.e., $BD2(i) = f(x_i, y_1)$ for $1 \le i \le NX$. f is u or u_y depending on whether BCTY(2) is 1 or 2. BD2 is not referenced when BCTY(2) = 3.
BD3	array of size NY containing boundary condition values for x = AX, i.e., $BD3(j) = f(x_1, y_j)$ for $1 \le j \le NY$. f is u or u_x depending on whether BCTY(3) is 1 or 2. BD3 is not referenced when BCTY(3) = 3.
BD4	array of size NX containing boundary condition values for y = BY, i.e., $BD4(i) = f(x_i, y_{NY})$ for $1 \le i \le NX$. f is u or u_y depending on whether BCTY(4) is 1 or 2. BD4 is not referenced when BCTY(4) = 3.

IORDER	order of accuracy of the discretization (2 or 4).
WORK	workspace array of size NWORK.
NWORK	length of the array WORK exactly as declared in the calling program (must be at least 5(NX + NY) + NX/2 + 15).

Output variables

U	on output, contains the solution at grid points, i.e., $U(i, j) = u(x_i, y_j)$ for $1 \le i \le NX$, $1 \le j \le NY$.
INFO	error flag. INFO = 0 indicates that the program ran to completion. INFO $= -k \neq 0$ $(1 \le k \le 8)$ indicates that error condition k was detected, INFO $= k \neq 0$ $(1 \le k \le 2)$ indicates warning k.

The output variable INFO is used to report errors and warnings. INFO is returned less than zero when an error is detected in the parameters passed by the user; a solution is not attempted in this case. INFO is returned greater than zero to alert the user of possible problems with the computed solution.

When $\lambda = 0$ and only periodic or Neumann boundary conditions are prescribed, the problem admits no solution unless the right side satisfies a consistency condition [13]. HFFT ensures that this condition is satisfied for the discrete problem by subtracting an appropriate constant from the right side. The constant is returned in WORK(1); if it is large in magnitude then the problem may have been posed incorrectly. The solution of the perturbed problem is a solution to the original discrete problem in the least squares sense [16]. Finally, the solution in this case is unique only up to an additive constant; HFFT returns the solution with minimum Euclidean norm and sets INFO = 2.

When $\lambda > 0$, a solution may not exist if λ is an eigenvalue of the Laplacian. If λ is near one of these values, then the problem may be ill-conditioned; in this case the computed solution may be grossly inaccurate because of significant round-off errors. HFFT returns INFO = 1 whenever $\lambda > 0$.

Note that there is no significant savings in storage when using HFFT2A or HFFT3A instead of HFFT2 or HFFT3. Although users of the latter two routines do not provide the arrays GH, BD1, BD2, and so on, the size of working storage is correspondingly larger.

2.2 Implementation Details

The basic components of the package are (a) drivers, (b) discretization modules, and (c) matrix decomposition modules. The drivers are the routines HFFT3, HFFT3A, HFFT2, and HFFT2A described above. The discretization modules compute the coefficients of a compact finite difference (HODIE) stencil [12] and form the right side of the resulting system of linear algebraic equations. Matrix decomposition modules solve this system using the Fourier algorithm [3]. A high-level depiction of this organization is given in Figure 1.

The entire HFFT package is coded in ANSI standard Fortran (1977); this has been verified using the ANSI option of the CDC FTN5 compiler [7] (version 5.1 running under NOS 2.1). Machine-dependent constants are taken from the PORT routine R1MACH [8]. The code has been tested on a CDC Cyber 180/855 and a Sun 3.

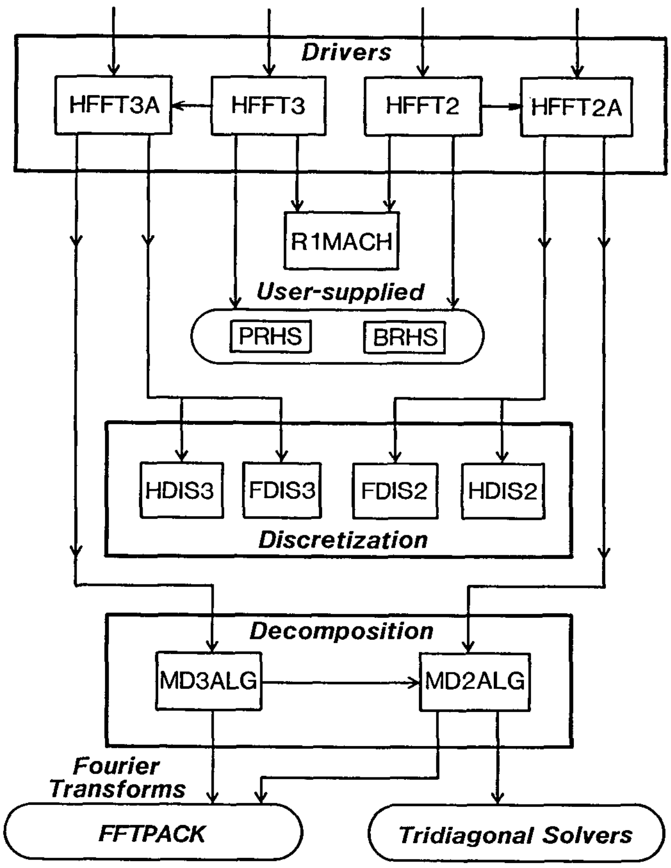

Fig. 1. Subprogram call graph for HFFT package. (Arrows emanate from calling subprogram and end at subprogram called.)

The discretization modules compute either a fourth-order accurate or second-order accurate HODIE discretization, as selected by the parameter IORDER. In the three-dimensional case the discretizations described in [2] (identified as methods A and B) are used, implemented by the routines HDIS3 and FDIS3, respectively. In two dimensions, discretizations obtained by restricting these to two dimensions are computed by the routines HDIS2 and FDIS2. Dirichlet boundary data are also loaded into the solution array at this time.

The Fourier method in three dimensions can be viewed as follows (for simplicity we assume an $n \times n \times n$ problem).

3D Algorithm

1. Fourier analysis in z direction (n^2 transforms of length n);
2. Solve n-independent two-dimensional problems, one for each z-plane;
3. Fourier synthesis in z direction (n^2 transforms of length n).

Thus the Fourier algorithm reduces the three-dimensional problem to a set of two-dimensional ones; in HFFT, the three-dimensional routine MDALG3 repeatedly calls the two-dimensional routine MDALG2. Each of the two-dimensional problems may also be solved by the Fourier algorithm.

2D Algorithm

1. Fourier analysis in y direction (n transforms of length n);
2. Solve n-independent one-dimensional problems, one for each x-line;
3. Fourier synthesis in y direction (n transforms of length n).

The one-dimensional problems require only the solution of a set of n symmetric tridiagonal systems of linear equations of order n. The exact form of these systems depends upon the boundary conditions in the x-direction—they have constant diagonals except for possibly the first and last elements of the main diagonal; in the periodic case they are Toeplitz systems. The form of these matrices is described in detail in [2].

To solve the tridiagonal systems we use an extension of the interlocking factorization method of Evans [5, 6], which is specific to diagonally dominant symmetric tridiagonal systems with constant diagonals. This algorithm requires approximately the same work as standard Gauss elimination for tridiagonal matrices, but has the advantage of requiring no extra storage; this is of interest when the algorithm is adapted to a vector computer (see Section 2.3). When $\lambda > 0$, the tridiagonal systems are not diagonally dominant and the Evans algorithm is unstable; in this case Gauss elimination with partial pivoting is used.

The exact form of the Fourier transform required, i.e., real periodic, sine, cosine, sine quarter-wave, and cosine quarter-wave, depends upon the boundary conditions. We use the FFTPACK software of Swarztrauber [17] in which these transforms are all provided. In addition, FFTPACK places no restriction on the length of the sequence being transformed, although the transform will only be "fast" if the length is a highly composite number. FFTPACK is based upon the autosort algorithms of Stockham as presented in [4].

2.3 Applicability to Vector Computers

The suitability of the Fourier method for vector computation depends principally upon the vectorization of the FFT and tridiagonal solution algorithms. The HFFT package has been modularized so that these components can easily be replaced with vector algorithms to increase the efficiency of the program on such machines. In this section we comment on these issues briefly. A more complete survey of applicable methods for vector and parallel computers may be found in [14].

Swarztrauber has reported [17] that FFTPACK vectorizes well on the Cray-1, running some five to seven times faster than the CDC 7600 on sequences of length 32 to 128. Like the Cooley–Tukey algorithm, the Stockham FFT is only pseudovectorizable, however, in that vector lengths vary during the course of the computation, with minimum vector length about $n^{1/2}$; this algorithm, as a result, does not appear to be as attractive for the Cyber 205.

An alternative method of vectorization is possible when many sequences are simulataneously available to be transformed, as is the case when solving partial differential equations. By taking each step of an FFT algorithm and applying it

Table I. Output of Example 1

Grid	Max Error
$5 \times 5 \times 5$	3.320E-03
$9 \times 9 \times 9$	1.779E-04
$13 \times 13 \times 13$	3.481E-05
$17 \times 17 \times 17$	1.092E-05
$21 \times 21 \times 21$	4.445E-06
$25 \times 25 \times 25$	2.137E-06
$29 \times 29 \times 29$	1.154E-06

to each sequence, one obtains an algorithm that maintains a constant vector length throughout the computation (and is also easier to implement). This is quite attractive for the Cyber 205; Sweet has recently built software that provides several of the transforms of FFTPACK in multitransform form with both scalar and Cyber 205 versions [18].

The Evans algorithm, like other algorithms for tridiagonal systems, is inherently recursive. However, since multiple systems must be solved simultaneously, we may vectorize by applying each step to each problem to obtain a vectorized multisystem solver. The fact that the Evans algorithm requires no additional workspace is particularly important in this context, since a workspace of length n for a single system solver would yield a workspace of length n^2 for an n-system solver. We have vectorized the Evans algorithm in this way, obtaining a speedup of approximately 23 for solving 500 systems of size 500 on the Cyber 205, in comparison to its performance on the 205 in scalar mode.

3. EXAMPLE

We next give a complete example that demonstrates how the subprogram HFFT3 is used to solve a routine problem. (For a more complete analysis of the performance of the HFFT subprograms, see [2].) We wish to find a function $u(x, y, z)$ that satifies

$$\begin{array}{ll}
\Delta u - \pi^2 u = g(x, y, z) & 0 < x, y, z < 1 \\
u(0, y, z) = 1 & 0 < y, z < 1 \\
u_x(1, y, z) = ze^z + \cos(2\pi y) & 0 < y, z < 1 \\
u(x, 0, z) = u(x, 1, z) & 0 < x, z < 1 \\
u_z(x, y, 0) = x & 0 < x, y < 1 \\
u(x, y, 1) = e^x + x\cos(2\pi y) & 0 < x, y < 1
\end{array}$$

where

$$g(x, y, z) = (x^2 + z^2 - \pi^2)e^{xz} - 5\pi^2 x \cos(2\pi y).$$

The solution to this problem is $u(x, y, z) = \exp(xz) + x\cos(2\pi y)$. The following Fortran program solves this problem using HFFT3, producing the results in Table I when run on a Cyber 180/855 computer compiled by the FTN5 compiler with OPT = 2.

```
C
C  EXAMPLE OF HFFT3 USAGE
C
C  ------------------------------------------------------------------------
C
C  PROBLEM   UXX + UYY + UZZ - PI**2*U = G(X, Y)
C
C            U = 1                          ON X = 0
C            UX = Z*EXP(Z) + COS(2*PI*Y)    ON X = 1
C            UZ = X                         ON Z = 0
C            U = EXP(X) + X*COS(2*PI*Y)     ON Z = 1
C
C            U(X, O, Z) = U(X, 1, Z)
C
C            G = (X**2 + Z**2 - PI**2)*EXP(X*Z) - 5*PI**2*X*COS(2*PI*Y)
C
C  SOLUTION  U = EXP(X*Z) + X*COS(2*PI*Y)
C
C  ------------------------------------------------------------------------
C
C  ----------------------
C  DECLARATIONS
C  ----------------------
C
C  ... CONSTANTS
C
C     PARAMETER (NMAX = 29)
      PARAMETER (NWORK = 3 + NMAX*(7 + NMAX*(11 + NMAX)))
      PARAMETER (LDXU = NMAX + 2)
C
C  ... GLOBAL VARIABLES
C
      COMMON /GLOBAL/ TWOPI, PISQR
      EXTERNAL PRHS, BRHS
C
C  ... LOCAL VARIABLES
C
      INTEGER BCTY(6)
      REAL U(LDXU, LDXU, LDXU), WORK(NWORK)
C
C  ------------------------------------------------------------------------
C
      PI = 4.0*ATAN(1.0)
      TWOPI = 2.0*PI
      PISQR = PI*PI
C
C  ------------------------
C  SETUP PROBLEM
C  ------------------------
C
      AX = 0.0
      BX = 1.0
      AY = 0.0
      BY = 1.0
      AZ = 0.0
      BZ = 1.0
      COEFU = -PISQR
      BCTY(1) = 2
      BCTY(2) = 3
      BCTY(3) = 1
      BCTY(4) = 3
      BCTY(5) = 1
      BCTY(6) = 2
      IORDER = 4
C
C  -------------------------------------------
C  SOLVE ON SEQUENCE OF GRIDS
C  -------------------------------------------
C
          PRINT 2000
          DO 500 N = 5, 29, 4
            NX = N
            NY = N
            NZ = N
C
C           ... SOLVE PDE
C
            CALL HFFT3(COEFU, PRHS, BRHS, AX, BX, AY, BY, AZ, BZ, NX,
     *                       NY, NZ, BCTY, IORDER, U, LDXU, LDXU, WORK,
     *                      NWORK, INFO)
            IF (INFO .LT. 0) GO TO 900
C
C           ... EVALUATE ERROR
C
            H = (BX - AX)/REAL)NX - 1)
            ERRMAX = 0.0E0
            DO 100 K = 1, NZ
                Z = AZ + REAL(K - 1)*H
                DO 100 J = 1, NY
                    Y = AY + REAL(J - 1)*H
                    DO 100 I = 1, NX
                        X = AX + REAL(I - 1)*H
                        TRUSOL = TRUE(X, Y, Z)
                        ERROR = ABS(TRUSOL - U(I, J, K))
                        ERRMAX = MAX(ERROR, ERRMAX)
  100       CONTINUE
            PRINT 2001, N, ERRMAX
  500     CONTINUE
          STOP
C
C         ... ERROR EXIT
C
  900     CONTINUE
          PRINT 2002, INFO
          STOP
C
 2000     FORMAT(/' GRID MAX-ERROR' /' ------------------' /)
 2001     FORMAT(4X, I2, 4X, 1PE10.3)
 2002     FORMAT(/' HFFT3 RETURNED INFO = ',I2)
C
          END
          FUNCTION PRHS (X, Y, Z)
```

```
C
C         ... RIGHT HAND SIDE OF PDE (USER-SUPPLIED)
C
          COMMON /GLOBAL/ TWOPI, PISQR
          PRHS = (X*X + Z*Z - PISQR)*EXP(X*Z)
     *           *5.0*PISQR*X*COS(TWOPI*Y)
          RETURN
          END
          FUNCTION BRHS(K, X, Y, Z)
C
C         ... RIGHT HAND SIDE OF BOUNDARY CONDITIONS (USER-
C            SUPPLIED)
C
          COMMON /GLOBAL/ TWOPI, PISQR
          GO TO (1, 2, 3, 4, 5, 6), K
          GO TO 999
C
        1 CONTINUE
          BRHS = Z*EXP(Z) + COS(TWOPI*Y)
          GO TO 999
C
        2 CONTINUE
          GO TO 999
C
        3 CONTINUE
          BRHS = 1.0
          GO TO 999
C
        4 CONTINUE
          GO TO 999
C
        5 CONTINUE
          BRHS = EXP(X) + X*COS(TWOPI*Y)
          GO TO 999
C
        6 CONTINUE
          BRHS = X
          GO TO 999
C
      999 CONTINUE
          RETURN
          END
          FUNCTION TRUE (X, Y, Z)
          COMMON /GLOBAL/ TWOPI, PISQR
          TRUE = EXP(X*Z) + X*COS(TWOPI*Y)
          RETURN
          END
```

4. ALGORITHM CONTENT

The following items are distributed with HFFT.

(1) HFFT3 (requires items 2 and 3)
(2) HFFT2 (requires item 3)
(3) Auxiliary software: FFTPACK [17] and R1MACH [8].

(4) Example of Section 3 (requires items 1, 2, and 3)
(5) Test program for HFFT3 (requires items 1, 2, and 3)
(6) Test program for HFFT2 (requires items 2 and 3)
(7) Test program for HW3CRT (requires item 3)
(8) Test program for SEPX4

The first two items contain the three-dimensional and two-dimensional software, respectively. The third item, while not formally part of this algorithm, contains software used by HFFT. The test programs, items five and six, each exercise the software on a battery of problems, including all test problems used in the companion paper [2]. Finally, programs that exercise the FISHPAK software HW3CRT and SEPX4 on the same problem sets as items 5 and 6, respectively, are included. These are made available to facilitate comparison of results with [2]. The HW3CRT and SEPX4 software are included in these files.

ACKNOWLEDGMENTS

Joan Baumann and John Nestor provided programming assistance in various phases of the construction of HFFT. The Fourier transforms in HFFT are performed by FFTPACK software, which was developed at the National Center for Atmospheric Research.

Disclaimer

Certain proprietary products have been referenced in this paper in order to fully describe the testing and implementation of HFFT software. Identification of such products does not imply recommendation or endorsement by the National Bureau of Standards.

REFERENCES

1. ADAMS, J., SWARZTRAUBER, P. N., AND SWEET, R. A. FISHPAK, a package of Fortran subprograms for the solution of separable elliptic partial differential equations. Version 3.1. 1981. NCAR Program Library, National Center for Atmospheric Research, P. O. Box 3000, Boulder, CO 80307.
2. BOISVERT, R. F. A fourth-order-accurate Fourier method for the Helmholtz equation in three dimensions. 1985. *ACM Trans. Math. Softw. 13,* 3 (Sept. 1987), 221–234.
3. BUZBEE, B. L., GOLUB, G. H., AND NIELSON, C. W. On direct methods for solving Poisson's equations. *SIAM J. Num. Anal. 7* (1970), 627–655.
4. COCHRAN, W. T. What is the fast Fourier transform? *IEEE Trans. Audio Electroacoustics 15* (1967), 45–55.
5. EVANS, D. J. An algorithm for the solution of certain tridiagonal systems of linear equations. *Computer J. 15* (1972), 356–359.
6. EVANS, D. J. Fast ADI methods for the solution of linear parabolic partial differential equations involving 2 space dimensions. *BIT 17* (1977), 486–491.
7. *Fortran Version 5 Reference Manual.* Control Data Corporation, CDC Publications and Graphics Division, P. O. Box 3492, Sunnyvale, CA 94088-3492, 1983.
8. FOX, P. A., HALL, A. D., AND SCHRYER, N. L. Algorithm 528: Framework for a portable library. *ACM Trans. Math. Softw. 4* (1978), 177–188.
9. HOUSTIS, E. N., AND PAPATHEODOROU, T. S. High-order fast elliptic equation solver. *ACM Trans. Math. Softw. 5* (1979), 431–441.
10. LYNCH, R. E. $O(h^4)$ and $O(h^6)$ finite difference approximations to the Helmholtz equation in n-dimensions. In *Advances in Computer Methods for Partial Differential Equations V,*

R. Vichnevetsky and R. S. Stepleman, Eds. IMACS, Rutgers Univ., New Brunswick, N.J., 1984, pp. 199–202.

11. LYNCH, R. E. *$O(h^6)$ Accurate Finite Difference Approximation to Solutions of the Poisson Equation in Three Variables.* CSD-TR 221, Computer Sciences Dept., Purdue Univ., West Lafayette, In., 1977.
12. LYNCH, R. E., AND RICE, J. R. High accuracy finite difference approximations to solutions of elliptic partial differential equations. *Proc. Nat. Acad. Sci. 75* (1978), 2541–2544.
13. MIKHLIN, S. G., ED. *Linear Equations of Mathematical Physics.* Holt, Reinhardt, and Winston, New York, 1967.
14. ORTEGA, J. M., AND VOIGT, R. G. Solution of partial differential equations on vector and parallel computers. *SIAM Rev. 17* (1985), 149–240.
15. RICE, J. R. AND BOISVERT, R. F. *Solving Elliptic Problems Using ELLPACK.* Springer-Verlag, New York, 1985.
16. SWARZTRAUBER, P. N., AND SWEET, R. *Efficient FORTRAN Subprograms for the Solution of Elliptic Partial Differential Equations.* NCAR Tech. Note IA-109, National Center for Atmospheric Research, Boulder, Co., 1975.
17. SWARZTRAUBER, P. N. Vectorizing the FFTs. In *Parallel Computation*, G. Rodrigue, Ed. Academic Press, New York, 1982, pp. 51–84.
18. SWEET, R. A. Fast Fourier transforms on a staggered grid. 1985. To appear.

Received January 1986; revised April 1987; accepted April 1987