The Fast Staggered Transform, composite symmetries, and compact symmetric algorithms

by B. L. Bradford

The Fast Staggered Transform (FST) is a variant of the fast Fourier transform (FFT) and is introduced to simplify and unify Fourier methods for the Poisson equation with boundary conditions specified on a staggered grid—one for which the boundary of the computational domain does not coincide with grid points, but is staggered at half grid spacings. Composite symmetric extensions of the computational domain are introduced for cases in which the boundary conditions are nonsymmetric. For example, one boundary may coincide with grid points while the opposite boundary is staggered. This is referred to as a mixed grid. Compact symmetric FFT and FST algorithms are a relatively new family of algorithms which offer significant performance improvements compared to traditional pre- and post-processing algorithms. The results of performance tests of both types of algorithms are presented. Furthermore, compact symmetric algorithms make possible the application of Fourier methods to six mixed grid boundary conditions which previously could not be treated by Fourier methods.

1. Introduction

We briefly review the Fourier analysis method for the Poisson equation on a rectangular region. For simplicity, we present this material in one spatial dimension, while in practice it is used in two or more spatial dimensions. In one spatial dimension, the discretized Poisson equation is

$$u_{n-1} - 2u_n + u_{n+1} = f_n$$

Table 1 Discrete homogeneous boundary conditions.

Acronym	*Boundary condition*	*Discrete analog*
C	Cyclic	$u_0 = u_M$
D	Dirichlet	$u_0 = 0$
N	Neumann	$u_2 - u_0 = 0$
DS	Dirichlet–staggered	$u_1 + u_0 = 0$
NS	Neumann–staggered	$u_1 - u_0 = 0$

for $1 \le n \le M$. We must specify boundary conditions at both the left and right endpoints. We may assume, without loss of generality, that the boundary conditions are homogeneous, since inhomogeneous boundary values may be absorbed into f_1 and f_M. The discrete, homogeneous boundary conditions we consider, specified for $n = 1$, are shown in **Table 1**. Note that we consider two variants of Dirichlet and Neumann boundary conditions, depending upon whether the boundary coincides with a grid point or is staggered at a half grid spacing. The notation D–N indicates a homogeneous Dirichlet boundary condition at the left endpoint and a homogeneous Neumann boundary condition at the right endpoint. Similar notation is used for other combinations. Combinations which involve only C, D, or N are referred to as standard grid boundary conditions. Combinations which involve only DS or NS are referred to as staggered grid boundary conditions. Other combinations are referred to as mixed grid boundary conditions.

The discretized boundary value problem may be written in matrix form as

$$Au = f, \tag{1}$$

where A is a matrix of dimension M, and u, f are vectors of length M. The boundary conditions have been used to eliminate u_0 and u_{M+1}. A is tridiagonal, and in one spatial dimension we would simply solve this linear system by Gaussian elimination.

However, we proceed as follows in anticipation of extensions to higher dimensions. First, we find the eigenvalues and eigenvectors of A. These are summarized in **Tables 2**, **3**, and **4**. Note that A always has a full set of linearly independent eigenvectors whose components are trigonometric expressions. Note also that in these tables the computational domain is different for each boundary condition. The reason for this will become clear after the corresponding symmetric FFT or FST has been studied. For this general discussion, we denote the eigenvalues by λ_k (repeated to multiplicity) and the corresponding eigenvectors by ϕ_k for $1 \le k \le M$. We now seek a solution for u in the form of an eigenvector expansion:

$$u = \sum_{k=1}^{M} \hat{u}_k \phi_k . \tag{2}$$

This requires that we also express f as an eigenvector expansion:

$$f = \sum_{k=1}^{M} \hat{f}_k \phi_k . \tag{3}$$

Since f is known and the vectors ϕ_k are linearly independent, we may compute $\hat{f}_k$. As we will see shortly, $\hat{f}_k$ may be computed most efficiently by means of a symmetric FFT. Thus, this step is referred to as Fourier analysis. Substituting Equations (2) and (3) into Equation (1) yields

$$\sum_{k=1}^{M} \hat{f}_k \phi_k = A\left[\sum_{k=1}^{M} \hat{u}_k \phi_k\right]$$

$$= \sum_{k=1}^{M} \hat{u}_k \lambda_k \phi_k .$$

Since the vectors ϕ_k are linearly independent, we conclude that

$$\hat{u}_k \lambda_k = \hat{f}_k$$

for $1 \le k \le M$.

We may now compute $\hat{u}_k$, unless $\lambda_k = 0$. In this case, the compatibility condition $\hat{f}_k = 0$ must hold, and $\hat{u}_k$ is arbitrary. Thus, the solution for u is not unique in this case. This occurs for C–C, N–N, NS–NS, N–NS, and NS–N boundary conditions, and corresponds to the fact that the solutions to these problems are unique only up to an additive constant. Having determined $\hat{u}_k$, we may now compute u using the inverse of the corresponding symmetric FFT. This step is called Fourier synthesis.

To clarify the role played by symmetric FFTs, we now consider a specific case. Assume that homogeneous Dirichlet boundary conditions are specified at both endpoints. In this case,

$$\lambda_k = -4 \sin^2[\pi k/2(M + 1)]$$

and

the nth component of $\phi_k = \sin[2\pi kn/2(M + 1)]$

for $1 \le k, n \le M$. Thus, the eigenvector expansion for the nth component of f (denoted f_n) is

$$f_n = \sum_{k=1}^{M} \hat{f}_k \sin[2\pi kn/2(M + 1)]$$

Table 2 Eigenstructure for the standard grid.

Boundary conditions	*nth component of eigenvectors*	*Computational domains*
Transforms	*Associated eigenvalues*	*Eigenvector indices*
C–C	$\cos(2\pi kn/N)$	$0 \le n \le N-1$
R FFT	$-4\sin^2(\pi k/N)$	$0 \le k \le N/2$ or $0 \le k \le (N-1)/2$
	and	and
	$\sin(2\pi kn/N)$	$0 \le n \le N-1$
	$-4\sin^2(\pi k/N)$	$1 \le k \le N/2-1$ or $1 \le k \le (N-1)/2$
N–N	$\cos(2\pi kn/N)$	$0 \le n \le N/2$
RE FFT	$-4\sin^2(\pi k/N)$	$0 \le k \le N/2$
D–D	$\sin(2\pi kn/N)$	$1 \le n \le N/2-1$
RO FFT	$-4\sin^2(\pi k/N)$	$1 \le k \le N/2-1$
N–D	$\cos[2\pi n(2k+1)/N]$	$0 \le n \le N/4-1$
RE–O FFT	$-4\sin^2[\pi(2k+1)/N]$	$0 \le k \le N/4-1$
D–N	$\sin[2\pi n(2k-1)/N]$	$1 \le n \le N/4$
RO–E FFT	$-4\sin^2[\pi(2k-1)/N]$	$1 \le k \le N/4$

Table 3 Eigenstructure for the staggered grid.

Boundary conditions	*nth component of eigenvectors*	*Computational domains*
Transforms	*Associated eigenvalues*	*Eigenvector indices*
NS–NS	$\cos[\pi k(2n+1)/N]$	$0 \le n \le N/2-1$
RSE FST	$-4\sin^2(\pi k/N)$	$0 \le k \le N/2-1$
DS–DS	$\sin[\pi k(2n+1)/N]$	$0 \le n \le N/2-1$
RSO FST	$-4\sin^2(\pi k/N)$	$1 \le k \le N/2$
NS–DS	$\cos[\pi(2k+1)(2n+1)/N]$	$0 \le n \le N/4-1$
RSE–SO FST	$-4\sin^2[\pi(2k+1)/N]$	$0 \le k \le N/4-1$
DS–NS	$\sin[\pi(2k+1)(2n+1)/N]$	$0 \le n \le N/4-1$
RSO–SE FST	$-4\sin^2[\pi(2k+1)/N]$	$0 \le k \le N/4-1$

Table 4 Eigenstructure for the mixed grid.

Boundary conditions	*nth component of eigenvectors*	*Computational domains*
Transforms	*Associated eigenvalues* $N = 2(2M + 1)$	*Eigenvector indices*
N–NS	$\cos(4\pi kn/N)$	$0 \le n \le M$
RE–E FFT	$-4\sin^2(2\pi k/N)$	$0 \le k \le M$
N–DS	$\cos[2\pi n(2k + 1)/N]$	$0 \le n \le M$
RE–O FFT	$-4\sin^2[\pi(2k + 1)/N]$	$0 \le k \le M$
D–NS	$\sin[2\pi n(2k - 1)/N]$	$1 \le n \le M$
RO–E FFT	$-4\sin^2[\pi(2k - 1)/N]$	$1 \le k \le M$
D–DS	$\sin(4\pi kn/N)$	$1 \le n \le M$
RO–O FFT	$-4\sin^2(2\pi k/N)$	$1 \le k \le M$
NS–N	$\cos[2\pi k(2n + 1)/N]$	$0 \le n \le M$
RSE–SE FST	$-4\sin^2(2\pi k/N)$	$0 \le k \le M$
NS–D	$\cos[\pi(2k + 1)(2n + 1)/N]$	$0 \le n \le M - 1$
RSE–SO FST	$-4\sin^2[\pi(2k + 1)/N]$	$0 \le k \le M - 1$
DS–N	$\sin[\pi(2k + 1)(2n + 1)/N]$	$0 \le n \le M$
RSO–SE FST	$-4\sin^2[\pi(2k + 1)/N]$	$0 \le k \le M$
DS–D	$\sin[2\pi k(2n + 1)/N]$	$0 \le n \le M - 1$
RSO–SO FST	$-4\sin^2(2\pi k/N)$	$1 \le k \le M$

for $1 \le n \le M$. A fast algorithm for computing $\hat{f}_k$ is obtained as follows. We make an odd, periodic extension of the original data f_n (referred to as RO-symmetric of length N):

$$f_{N-n} = -f_n$$

and

$$f_{N+n} = f_n,$$

where $N = 2(M + 1)$. Note that

$$f_0 = 0$$

and

$$f_{N/2} = 0.$$

Thus, the original data f_n have been extended in a manner related to homogeneous Dirichlet boundary conditions. It happens that the sequence $\hat{f}_k$ is related to the discrete Fourier transform (DFT) of the extended sequence f_n. This relationship is given by

$$-1/2\, i\hat{f}_k = \text{DFT}(f_n) = 1/N \sum_{n=0}^{N-1} f_n \omega_N^{-kn}$$

for $0 \le k \le N - 1$, where

$$\omega_N = e^{i2\pi/N}.$$

It can be shown that the sequence $\hat{f}_k$ represented as a DFT is real, odd, and periodic of length N:

$$\hat{f}_{N-k} = -\hat{f}_k$$

and

$$\hat{f}_{N+k} = \hat{f}_k .$$

The importance of this relationship to the DFT is due to the fast Fourier transform (FFT) which is a fast algorithm for computing the DFT. Similarly, u_n may be computed from $\hat{u}_k$ by means of an inverse fast Fourier transform (IFFT). Thus, FFT algorithms are responsible for the efficiency of the eigenvector expansion method.

2. Fast staggered transform (FST)

Next, assume that the boundary points are staggered at a half grid spacing, and that homogeneous Neumann boundary conditions are specified at each of these. We refer to this as a Neumann–staggered boundary condition, which is approximated at $n = 0$ by $u_0 - u_{-1} = 0$. In this case,

$$\lambda_k = -4 \sin^2[\pi k/2(M + 1)]$$

and

the nth component of $\phi_k = \cos[\pi k(2n + 1)/2(M + 1)]$

for $0 \le k, n \le M$. Thus, the eigenvector expansion for the nth component of f is

$$f_n = 1/2\hat{f}_0 + \sum_{k=1}^{M} \hat{f}_k \cos[\pi k(2n + 1)/2(M + 1)]$$

for $0 \le n \le M$. A fast algorithm for computing $\hat{f}_k$ is obtained as follows. We make a staggered even, periodic extension of the original data f_n (referred to as RSE-symmetric of length N):

$$f_{N-n-1} = f_n$$

and

$$f_{N+n} = f_n ,$$

where $N = 2(M + 1)$. Note that

$$f_{N-1} = f_0$$

and

$$f_{N/2-1} = f_{N/2} .$$

Thus, the original data f_n have been extended in a manner related to homogeneous Neumann–staggered boundary conditions. It is shown in [1] that the sequence $\hat{f}_k$ is related to the discrete staggered transform (DST) of the extended sequence f_n. This relationship is given by

$$1/2\hat{f}_k = \mathrm{DST}(f_n) = 1/N \sum_{n=0}^{N-1} f_n \omega_N^{-k(n+1/2)}$$

for $0 \le k \le N - 1$. It can be shown that the sequence $\hat{f}_k$ represented as a DST is real, odd, and odd-periodic of length N:

$$\hat{f}_{N-k} = -\hat{f}_k$$

and

$$\hat{f}_{N+k} = -\hat{f}_k .$$

The importance of this relationship to the DST is due to the fast staggered transform (FST), which is a fast algorithm for computing the DST similar in nature to the FFT. A simple example of the FST algorithm is described in Section 6; more detail may be found in [1]. By means of an inverse fast staggered transform (IFST), u_n may be computed from $\hat{u}_k$.

3. Composite symmetries

We now modify this problem and analyze the impact on the solution method. Assume that the left boundary point is staggered at a half grid spacing, whereas the right boundary point coincides with a grid point, and that homogeneous Neumann boundary conditions are specified at each of these. We refer to these as NS–N boundary conditions (see Table 1). In this case,

$$\lambda_k = -4 \sin^2[\pi k/(2M + 1)]$$

and

the nth component of $\phi_k = \cos[\pi k(2n + 1)/(2M + 1)]$

for $0 \le k, n \le M$. Thus, the eigenvector expansion for the nth component of f is

$$f_n = 1/2\hat{f}_0 + \sum_{k=1}^{M} \hat{f}_k \cos[\pi k(2n + 1)/(2M + 1)]$$

for $0 \le n \le M$.

A fast algorithm for computing $\hat{f}_k$ is obtained by modifying that for NS–NS boundary conditions. We have already seen that a staggered even, periodic extension of the original data f_n (RSE symmetry) corresponds to NS–NS boundary conditions. However, in this case our boundary conditions are nonsymmetric. Our approach is to superimpose a second symmetry which corresponds to a Neumann boundary condition (not staggered) at the right endpoint. We refer to this as a composite symmetry. More specifically, we make a composite staggered even–staggered even, periodic extension of the original data f_n (referred to as RSE–SE-symmetric of length N):

$$f_{N-n-1} = f_n ,$$

$$f_{N/2-n-1} = f_n ,$$

and

$$f_{N+n} = f_n ,$$

Table 5 Symmetries in the IDFT.

Acronym	*Symmetry*	*Sequence*	*DFT*
	Periodic	$x_{N+n} = x_n$	$X_{N+k} = X_k$
R	Real	$\bar{x}_n = x_n$	$X_{N-k} = \bar{X}_k$
RE	Real Even	$\bar{x}_n = x_n$ $x_{N-n} = x_n$	$\bar{X}_k = X_k$ $X_{N-k} = X_k$
RO	Real Odd	$\bar{x}_n = x_n$ $x_{N-n} = -x_n$	$\bar{X}_k = -X_k$ $X_{N-k} = -X_k$
RE–E	Real composite Even–Even (N even)	$\bar{x}_n = x_n$ $x_{N-n} = x_n$ $x_{N/2-n} = x_n$	$\bar{X}_k = X_k$ $X_{N-k} = X_k$ $X_k = (-1)^k X_k$
RE–O	Real composite Even–Odd (N even)	$\bar{x}_n = x_n$ $x_{N-n} = x_n$ $x_{N/2-n} = -x_n$	$\bar{X}_k = X_k$ $X_{N-k} = X_k$ $X_k = (-1)^{k+1} X_k$
RO–E	Real composite Odd–Even (N even)	$\bar{x}_n = x_n$ $x_{N-n} = -x_n$ $x_{N/2-n} = x_n$	$\bar{X}_k = -X_k$ $X_{N-k} = -X_k$ $X_k = (-1)^{k+1} X_k$
RO–O	Real composite Odd–Odd (N even)	$\bar{x}_n = x_n$ $x_{N-n} = -x_n$ $x_{N/2-n} = -x_n$	$\bar{X}_k = -X_k$ $X_{N-k} = -X_k$ $X_k = (-1)^k X_k$

where $N = 2(2M + 1)$. Note that

$$f_{N-1} = f_0$$

and

$$f_{M-1} = f_{M+1}.$$

Thus, the original data f_n have been extended in a manner related to homogeneous NS–N boundary conditions. It is shown in [1] that the sequence $\hat{f}_k$ is related to the discrete staggered transform (DST) of the extended sequence f_n. This relationship is given by

$$1/2\hat{f}_k = F_{2k}$$

for $0 \le k \le M$, where F_{2k} are the even terms of the DST of f_n. It can be shown that the sequence F_k is real, odd, with zero odd terms, and odd-periodic of length N:

$$F_{N-k} = -F_k,$$

$$F_k = (-1)^k F_k,$$

and

$$F_{N+k} = -F_k.$$

As before, the nonzero values F_{2k} may be computed efficiently by means of the fast staggered transform (FST). Similarly, u_n may be computed from $\hat{u}_k$ by means of an inverse fast staggered transform (IFST).

Finally, we assume that the left boundary point is staggered at a half grid spacing with a homogeneous Neumann boundary condition specified, while the right boundary point coincides with a grid point with a homogeneous Dirichlet boundary condition specified. We refer to these as NS–D boundary conditions (see Table 1). In this case,

$$\lambda_k = -4 \sin^2[\pi(2k + 1)/(2(2M + 1))]$$

and

$$\text{the } n\text{th component of } \phi_k = \cos[\pi(2k + 1)(2n + 1)/(2(2M + 1))]$$

for $0 \le k, n \le M - 1$. Thus, the eigenvector expansion for the nth component of f is

Table 6 Symmetries in the IDST.

Acronym	*Symmetry*	*Sequence*	*DST*
	Periodic	$x_{N+n} = x_n$	$X_{N+k} = -X_k$
R	Real	$\bar{x}_n = x_n$	$X_{N-k} = -\bar{X}_k$
RSE	Real Staggered (S) Even	$\bar{x}_n = x_n$ $x_{N-n-1} = x_n$	$\bar{X}_k = X_k$ $X_{N-k} = -X_k$
RSO	Real Staggered (S) Odd	$\bar{x}_n = x_n$ $x_{N-n-1} = -x_n$	$\bar{X}_k = -X_k$ $X_{N-k} = X_k$
RSE–SE	Real composite S. Even–S. Even (N even)	$\bar{x}_n = x_n$ $x_{N-n-1} = x_n$ $x_{N/2-n-1} = x_n$	$\bar{X}_k = X_k$ $X_{N-k} = -X_k$ $X_k = (-1)^k X_k$
RSE–SO	Real composite S. Even–S. Odd (N even)	$\bar{x}_n = x_n$ $x_{N-n-1} = x_n$ $x_{N/2-n-1} = -x_n$	$\bar{X}_k = X_k$ $X_{N-k} = -X_k$ $X_k = (-1)^{k+1} X_k$
RSO–SE	Real composite S. Odd–S. Even (N even)	$\bar{x}_n = x_n$ $x_{N-n-1} = -x_n$ $x_{N/2-n-1} = x_n$	$\bar{X}_k = -X_k$ $X_{N-k} = X_k$ $X_k = (-1)^{k+1} X_k$
RSO–SO	Real composite S. Odd–S. Odd (N even)	$\bar{x}_n = x_n$ $x_{N-n-1} = -x_n$ $x_{N/2-n-1} = -x_n$	$\bar{X}_k = -X_k$ $X_{N-k} = X_k$ $X_k = (-1)^k X_k$

$$f_n = \sum_{k=0}^{M-1} \hat{f}_k \cos[\pi(2k + 1)(2n + 1)/(2(2M + 1))]$$

for $0 \le n \le M - 1$.

A fast algorithm for computing $\hat{f}_k$ is obtained by modifying the ideas above. As before, we superimpose a second symmetry which will correspond to a Dirichlet boundary condition (not staggered) at the right endpoint. More specifically, we make a composite staggered even–staggered odd, periodic extension of the original data f_n (referred to as RSE–SO-symmetric of length N):

$$f_{N-n-1} = f_n,$$

$$f_{N/2-n-1} = -f_n,$$

and

$$f_{N+n} = f_n,$$

where $N = 2(2M + 1)$. Note that

$$f_{N-1} = f_0$$

and

$$f_M = 0.$$

Thus, the original data f_n have been extended in a manner related to homogeneous NS–D boundary conditions. It is shown in [1] that the sequence $\hat{f}_k$ is related to the discrete staggered transform (DST) of the extended sequence f_n. This relationship is given by

$$1/2\hat{f}_k = F_{2k+1}$$

for $0 \le k \le M - 1$, where F_{2k+1} are the odd terms of the DST of f_n. It can be shown that the sequence F_k is real, odd, with zero even terms, and odd-periodic of length N:

$$F_{N-k} = -F_k,$$

$$F_k = (-1)^{k+1} F_k,$$

and

$$F_{N+k} = -F_k.$$

As before, the nonzero values F_{2k+1} may be computed efficiently by means of the fast staggered transform (FST). Similarly, u_n may be computed from $\hat{u}_k$ by means of an inverse fast staggered transform (IFST).

Using techniques similar to those illustrated above, we have developed Fourier methods for 17 boundary conditions for the Poisson equation. These are summarized briefly in tabular form. The 17 boundary conditions are listed in Tables 2–4. Beneath each boundary condition is the associated symmetric extension of the computational domain and the transform (DFT or DST). These symmetries are defined in **Tables 5** and **6**.

4. Overview of compact symmetric algorithms

It is clear from the preceding discussion that FFT algorithms (or variants thereof) form the core of fast eigenvector expansion methods. Therefore, we discuss a relatively new family of symmetric FFT algorithms which offer significant performance improvements and are applicable to a broader range of boundary conditions. Recall that in the eigenvector expansion method we found it useful to extend a given sequence of real data according to a symmetry which is related to the boundary conditions. We then computed the FFT of this real, symmetric sequence. These symmetries in the data result in many redundant computations in the complex FFT algorithm. Therefore, specialized FFT algorithms, referred to as symmetric FFTs, have been developed which eliminate these redundant computations.

The traditional symmetric FFT algorithms are referred to as pre- and post-processing algorithms. These algorithms are described in detail in [2, 3], and related information may be found in [4, 5]. We briefly summarize the idea behind these algorithms. Given a symmetric sequence x_n of length N, where N is even, we first compute an auxiliary sequence y_n of length $N/2$ which is a simple linear function of x_n. This is the pre-processing step. We then input y_n into a complex FFT algorithm and obtain its transform Y_k. The transform X_k of x_n is then recovered from Y_k. This is the post-processing step. The operation counts for the pre- and post-processing steps are of asymptotically lower order than that for the complex FFT of y_n. Since y_n has length $N/2$, the operation count for the pre- and post-processing algorithms is approximately half that of computing the complex FFT of the symmetric sequence x_n directly. However, for sequences of practical length, the low-order operation counts of the pre- and post-processing steps are still significant. Furthermore, the pre- and post-processing steps require additional data accesses which also contribute to the total execution time.

Compact symmetric FFTs are a relatively new family of symmetric FFTs which eliminate redundant computations due to symmetries in the data, and also eliminate the additional computations and data accesses associated with pre- and post-processing algorithms. Compact symmetric FFTs are also available for symmetric sequences of length N where N is odd. This has applications in fast Poisson solvers in which one boundary coincides with grid points, while the opposite boundary is staggered at a half grid spacing.

We now briefly review the development of compact symmetric FFTs. A compact symmetric FFT known as Edson's algorithm has long been available for real sequences. In [6], a compact symmetric FFT for real even sequences is introduced, but in the context of Clenshaw–Curtis quadrature. In [3], in-place compact symmetric FFTs are developed for real, even, odd, quarterwave even, and quarterwave odd symmetries. All in-place algorithms based on the splitting method require either the input or output sequence to be in a permuted order, referred to as bit-reversed order. These in-place algorithms require the input sequence in physical space to be in bit-reversed order, and produce the forward transform in natural order. From our discussion of the eigenvector expansion method, it is clear that this is the opposite of what is desired. In [7], analogous algorithms are developed which accept the input sequence in physical space in natural order, and produce the forward transform in bit-reversed order. We follow the general approach set forth in [7]. A simple modification of this approach using additional storage space leads to out-of-place implementations which allow both the input and output sequence to be in natural order. While this is not necessary for applications to fast Poisson solvers, it does result in a software interface which is easier to understand. The software whose performance is analyzed in Section 5 is an out-of-place implementation, and the additional storage space required does not exceed half of the cache size on the IBM RISC System/6000® (RS/6000) processor.

5. Performance of symmetric algorithms

Before discussing compact symmetric FFTs in more detail, we indicate the magnitude of the performance improvement offered by these algorithms in comparison to pre- and post-processing algorithms. In [1], software is designed and developed for the compact symmetric FFT for real, odd (RO) symmetric sequences (the sine transform). We now compare the performance of this software to an implementation of the pre- and post-processing algorithm for the sine transform which uses the real FFT from the IBM Engineering/Scientific Subroutine Library (ESSL) on the IBM RS/6000 processor. The FFT software in ESSL is very highly optimized for the RS/6000 processor, and in order to make a fair comparison of the performance of these two algorithms we used similar

Table 7 Timing data for 3D grid on the IBM RS/6000 processor.

N	*CINIT*	*CTRAN*	*PINIT*	*PTRAN*	*DELTIM*
32	0.000755	0.002703	0.000181	0.005246	48.5
64	0.001540	0.023219	0.000108	0.034237	32.2
128	0.006136	0.209906	0.000189	0.295097	28.9
256	0.022428	2.105508	0.000358	2.778550	24.2

N length of the RO symmetric sequence
CINIT compact algorithm initialization time (seconds)
CTRAN compact algorithm transform time (seconds)
PINIT pre- and post-processing algorithm initialization time (seconds)
PTRAN pre- and post-processing algorithm transform time (seconds)
DELTIM 100(PTRAN − CTRAN)/PTRAN

coding techniques for the compact algorithm as well as the pre- and post-processing steps. The most important issues are cache management and facilitating the use of the floating-point multiply/add (FMA) instruction.[1] Our performance results are for the forward and inverse transform of a three-dimensional grid of size $(N/2)^3$ (N is the length of the RO-symmetric sequence, and the computational domain is $1 \le n \le N/2 - 1$). That is, we perform two transforms of $(N/2)^2$ RO-symmetric sequences of length N. The sequences being transformed are stored with a stride of one (column ordering in FORTRAN), and they are processed in blocks which fit within the cache size (64KB for the RS/6000 Model 550 processor). Assembly language listings were reviewed to ensure that the FMA (and FMS) instruction was used by the compiler whenever possible. The timing data are summarized in **Table 7**.

6. Details of compact symmetric algorithms

Software for compact symmetric FFTs and FSTs is significantly more complicated than that for pre- and post-processing algorithms. To give an idea of what is involved, we now provide a high-level description of the compact symmetric FST for real, staggered even (RSE)-symmetric sequences of length N, where N is a power of 2 (referred to as radix = 2). A more general algorithm for any composite value of N is developed in detail in [1]. Given an RSE-symmetric sequence x_n of length N, its DST X_k is real (R) and odd conjugate symmetric (OCS). These symmetries are defined in **Table 8**. We indicate that X_k satisfies both of these symmetries by noting that X_k is ROCS-symmetric. We develop an algorithm for computing the IDST of X_k which requires that X_k be input in a permuted order. The output of this algorithm is x_n in natural order. By algebraically inverting this algorithm for the IDST, we obtain an algorithm for computing the DST of x_n which allows x_n to be input in natural order.

[1] See [8–10] for the development of FFT algorithms for machines with fused multiply/add instructions. Such algorithms might also be used as a starting point for the developments in this paper and are an area for further research. A real-input FFT for multiply/add machines is discussed explicitly in [9].

The ROCS-symmetric sequence X_k is split into two subsequences, consisting of the even-numbered terms and the odd-numbered terms. The subsequence of even terms is ROCS-symmetric, while the subsequence of odd terms is RSOCS-symmetric (see Table 8), both with length $N/2$. Assume for the moment that we know the IDST of these subsequences. We develop equations, referred to as combine equations, for obtaining the IDST of the parent sequence from the IDST of its subsequences. These combine equations are not developed here, but may be found in [1]. The IDSTs of the subsequences are obtained by splitting them and repeating the process. Eventually, subsequences of length 1 are obtained, for which computing the IDST is trivial. This completes the algorithm.

Each time a sequence is split, the resulting subsequences satisfy certain symmetries. This information is recorded in the "splitting tree" in **Figure 1**. Refer to Table 8 for definitions of the symmetries appearing in the splitting tree. In the tables, p refers to a factor of N, while q identifies subsequences and satisfies $0 \le q \le p - 1$. At each stage of the algorithm, we must fully exploit these symmetries in order to eliminate redundant data and computations. In a sense, the entire algorithm is embodied in the splitting tree. For reasonably general composite values of N, the splitting tree becomes significantly more complicated than the example shown in Figure 1. In fact, the software implementation of this algorithm must be capable of automatically generating the splitting tree for a given value of N, and then "reading" this splitting tree in order to execute the algorithm correctly. This has been accomplished by providing the software with a knowledge base which describes the symmetries of the subsequences resulting from the splitting of a symmetric parent sequence. This knowledge base must include all symmetries which appear at any stage of the algorithm, and all factors of N which are allowed (usually 2, 3, 5, 7).

It should be noted that the use of composite factors of N may lead to more efficient implementations.[2] For example, the software whose performance is analyzed in Section 5 uses factors of 4 and 8 rather than 2. Each factor processed results in an access of all of the data. Thus, larger factors result in fewer data accesses provided that the target hardware has a sufficient number of floating-point registers to process each factor without storing intermediate results. The IBM RS/6000 processor, having 32 floating-point registers, is capable of processing factors larger than 8, but we have not yet implemented this.

The compact algorithms for composite symmetries are special cases of those for the more basic even and odd symmetries. As an example, we now describe the compact

[2] See [4, 8] for a performance analysis of the use of composite factors of N.

Table 8 Symmetries in the DST.

Acronym	*Symmetry*	*Sequence*	*IDST*
	Periodic	$X_{N+k} = -X_k$	$x_{N+n} = x_n$
OCS	Odd Conj Sym	$X_{N-k} = -\bar{X}_k$	$\bar{x}_n = x_n$
SOCS	Stag Odd Conj Sym	$X_{N-k-1} = -\bar{X}_k$	$x_n = \omega_N^{-(n+1/2)}\bar{x}_n$ $x_n = \omega_N^{-(n+1/2)/2}\bar{x}_n$
OCSIS	OCS Indcd Interseq Sym	$X_{k,p-q} = -\bar{X}_{N/p-k-1,q}$	$y_{n,p-q} = \omega_{N/p}^{-(n+1/2)}\bar{y}_{n,q}$
SOCSIS	SOCS Indcd Interseq Sym	$X_{k,p-q-1} = -\bar{X}_{N/p-k-1,q}$	$y_{n,p-q-1} = \omega_{N/p}^{-(n+1/2)}\bar{y}_{n,q}$
R	Real	$\bar{X}_k = X_k$	$x_{N-n-1} = \bar{x}_n$
I	Imag	$\bar{X}_k = -X_k$	$x_{N-n-1} = -\bar{x}_n$
ROCSZO	ROCS & zero Odd terms (N even)	$\bar{X}_k = X_k$ $X_{N-k} = -X_k$ $X_k = (-1)^k X_k$	$\bar{x}_n = x_n$ $x_{N-n-1} = x_n$ $x_{N/2-n-1} = x_n$
ROCSZE	ROCS & zero Even terms (N even)	$\bar{X}_k = X_k$ $X_{N-k} = -X_k$ $X_k = (-1)^{k+1} X_k$	$\bar{x}_n = x_n$ $x_{N-n-1} = x_n$ $x_{N/2-n-1} = -x_n$
IOCSZE	IOCS & zero Even terms (N even)	$\bar{X}_k = -X_k$ $X_{N-k} = X_k$ $X_k = (-1)^{k+1} X_k$	$\bar{x}_n = x_n$ $x_{N-n-1} = -x_n$ $x_{N/2-n-1} = x_n$
IOCSZO	IOCS & zero Odd terms (N even)	$\bar{X}_k = -X_k$ $X_{N-k} = X_k$ $X_k = (-1)^k X_k$	$\bar{x}_n = x_n$ $x_{N-n-1} = -x_n$ $x_{N/2-n-1} = -x_n$
Z	Zero	$X_k = 0$	$x_n = 0$

algorithms for RSE–SE- and RSE–SO-symmetric FSTs. Given an RSE–SE-symmetric sequence x_n of length $N = 2(2M + 1)$, its DST X_k is real (R), odd conjugate symmetric (OCS), with zero odd terms (ZO). We indicate that X_k satisfies all of these symmetries by noting that X_k is ROCSZO-symmetric. As before, we develop an algorithm for computing the IDST of X_k. The ROCSZO-symmetric sequence X_k is split into two subsequences consisting of the even-numbered terms and the odd-numbered terms. The subsequence of even terms is ROCS-symmetric, while

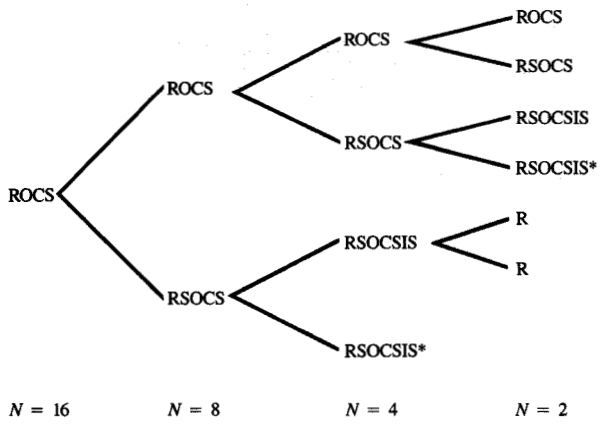

Figure 1

Splitting tree for radix-2 RSE FST with $N = 16$. In Figures 1–3, an asterisk indicates a subsequence which is redundant because of an intersequence symmetry.

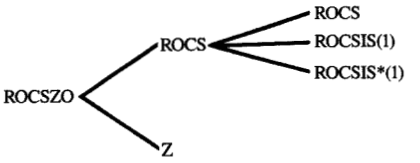

Figure 2

Splitting tree for RSE–SE-symmetric FST.

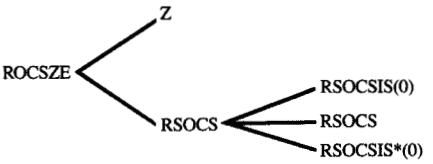

Figure 3

Splitting tree for RSE–SO-symmetric FST.

the subsequence of odd terms is Z-symmetric (identically zero), both with length $2M + 1$. The IDST of a Z sequence is also a Z sequence, so this branch of the splitting tree requires no further processing. We continue splitting the other branch of the splitting tree as before. However, we note that N contains only one factor of 2, so all remaining factors are odd. As an example, **Figure 2** shows the splitting tree for the first two factors of N, these being 2 and 3, respectively.

Recall that the RSE–SE-symmetric FST is associated with NS–N boundary conditions. There is an alternative approach for NS–N boundary conditions which highlights one of the advantages of compact algorithms over pre- and post-processing algorithms. From Figure 2, it is clear that we could have started with an RSE-symmetric sequence of length $2M + 1$ instead. Pre- and post-processing algorithms have the restriction that the length of a symmetric sequence must be even, so a compact algorithm is required to proceed further. We have chosen the approach of composite symmetries because it leads to a consistent approach for all nonsymmetric boundary conditions. For example, NS–D boundary conditions are associated with the RSE–SO FST. Given an RSE–SO-symmetric sequence x_n of length $N = 2(2M + 1)$, its DST X_k is real (R), odd conjugate symmetric (OCS), with zero even terms (ZE). We indicate that X_k satisfies all of these symmetries by noting that X_k is ROCSZE-symmetric. The remainder of the algorithm for the RSE–SO FST is analogous to that for the RSE–SE FST, and its splitting tree is shown in **Figure 3**.

7. Estimate of software package size

As is often the case with mathematical algorithms, there are some disadvantages as well as advantages associated with compact symmetric FFTs and FSTs. Pre- and post-processing algorithms require only a modest quantity of simple, straightforward software to supplement a library of real and complex FFTs. On the other hand, software for compact algorithms is complex, and unique software is required for each of the basic transforms. We now estimate the quantity of code required to implement all of the compact symmetric FFTs and FSTs developed in [1]. There it was shown that 17 boundary conditions for the Poisson equation can be addressed by five basic transforms (R, RE, RO FFTs and RSE, RSO FSTs). By a basic transform we mean both the forward and inverse directions, since one direction is seldom useful without the other. For each basic transform, we have identified a need

for four factors of the sequence length N, namely 2, 3, 5, 7 (for simplicity, we are ignoring composite factors, as discussed in Section 6). If we combine the independent options discussed above, we obtain at least $5 \times 4 = 20$ variations of the basic transforms. From prototype software for the RO FFT, we estimate that each variation requires approximately 1800 lines of FORTRAN code (including comments). Thus, the entire package of 20 variations requires approximately 36K lines of FORTRAN code.

However, there is a great deal of similarity between the codes for these compact symmetric algorithms. More importantly, in [1] it is shown how a symbolic computational tool such as Mathematica [11] can be used to automate the most labor-intensive steps in the implementation process. Specifically, Mathematica can be used to algebraically simplify the general combine equations for specific factors of the sequence length N, and map the inputs and outputs to storage locations which allow the algorithm to be implemented in place. The Mathematica output can be inserted into a FORTRAN subroutine skeleton, compiled, and executed. Performance tuning is still required for the target hardware.

8. Summary

For each of the boundary conditions in Tables 2, 3, and 4, an FFT or FST algorithm has been developed which computes the coefficients in the corresponding eigenvector expansion as efficiently as possible by eliminating all redundant computations which would occur in the full complex FFT, and without pre- or post-processing. We conclude by summarizing how the algorithms presented at a high level in this paper, and in much greater detail in [1], are related to previously published algorithms.

The algorithms in [7] were developed for radix-2 only. We have generalized all of these to radix-p for a general factor p, yielding mixed radix algorithms.[3] This has resulted in a number of new intermediate symmetries which occur in the course of the splitting method. After the combine equations are obtained for the inverse transform, they must be inverted to obtain those for the forward transform. For the radix-p algorithms, this requires the inversion of many systems of p equations in p unknowns. We have exploited the special nature of these systems of equations to invert them in closed form. The real quarterwave even and quarterwave odd transforms, which we refer to as the real staggered even (RSE) and real staggered odd (RSO) FFTs, have been used for N–D and D–N boundary conditions, respectively. We have shown that the algorithms for these symmetries in [7] are not in place. We have developed two new compact symmetric FFTs, called real composite even-odd (RE–O) and composite odd-even (RO–E) for these boundary conditions. We have shown that these new algorithms are in place and obtain the goal of eliminating all redundant operations which would occur in the full complex FFT.

For staggered grid boundary conditions, we have developed new algorithms based on a variant of the DFT which we refer to as the discrete staggered transform (DST). In analogy with the FFT, we have developed efficient algorithms for computing the DST, which we refer to as the fast staggered transform (FST). Previously, the only known algorithms for staggered grid boundary conditions were the real quarterwave even and quarterwave odd FFTs, and the pre- and post-processing algorithms in [13]. Although the real quarterwave even and quarterwave odd FFTs have been used for NS–NS and DS–DS boundary conditions, respectively, the algorithms for these symmetries in [7] are not in place. The pre- and post-processing algorithms for NS–DS and DS–NS boundary conditions are less efficient than the new compact symmetric FSTs for the same general reasons discussed previously.

For mixed grid boundary conditions, we have developed new algorithms based on superimposing two symmetries. We refer to the resulting symmetries as composite symmetries. Previously, the only known algorithms for mixed grid boundary conditions were the pre- and post-processing algorithms in [13] for NS–D and D–NS boundary conditions. Again, the pre- and post-processing algorithms are less efficient than the new compact algorithms. Furthermore, we have developed compact algorithms for six mixed grid boundary conditions which previously could not be treated by Fourier methods.

[3] Related work is included in [4, 5]. The Good–Thomas FFT [4, 12] might also be used as a starting point for these developments, and may have advantages for the mixed radix case. This is an area for further research.

Acknowledgments

This work was generously supported by the IBM Federal Systems Company Resident Study Program. Larry Carter (Mathematical Sciences Department, IBM Research Division, Yorktown Heights, NY) provided suggestions for optimizing the software for the compact RO FFT for the IBM RS/6000 processor.

References

1. B. L. Bradford, "Fast Fourier Transforms for Direct Solution of Poisson's Equation," Ph.D. thesis, University of Colorado at Denver, 1991.
2. J. W. Cooley, P. A. W. Lewis, and P. D. Welsh, "The Fast Fourier Transform Algorithm: Programming Considerations in the Calculation of Sine, Cosine and Laplace Transforms," *J. Sound Vibration* **12**, 315–337 (1970).
3. P. N. Swarztrauber, "Symmetric FFTs," *Math. Comp.* **47**, No. 175, 323–346 (1986).

4. R. E. Blahut, *Fast Algorithms for Digital Signal Processing*, Addison-Wesley Publishing Co., Reading, MA, 1986.
5. P. Duhamel, "Implementation of 'Split-Radix' FFT Algorithms for Complex, Real, and Real-Symmetric Data," *IEEE Trans. Acoust., Speech, Signal Process.* **34**, 285–295 (1986).
6. W. M. Gentleman, "Implementing Clenshaw–Curtis Quadrature: II. Computing the Cosine Transformation," *Commun. ACM* **15**, No. 5, 343–346 (1972).
7. W. L. Briggs, "Further Symmetries of In-Place FFTs," *SIAM J. Sci. Statist. Comput.* **8**, No. 4, 644–654 (1987).
8. E. Linzer and E. Feig, "Implementation of Efficient FFT Algorithms on Fused Multiply/Add Architectures," *IEEE Trans. Signal Process.* **41**, No. 1, 93–107 (1993).
9. E. Linzer and E. Feig, "Modified FFTs for Fused Multiply/Add Architectures," *Math. Comp.* **60**, No. 201, 347–361 (1993).
10. C. Lu, J. W. Cooley, and R. Tolimieri, "Variants of the Winograd Multiplicative FFT Algorithms and Their Implementation on IBM RS/6000," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1991, pp. 2185–2188.
11. S. Wolfram, *Mathematica: A System for Doing Mathematics by Computer*, Addison-Wesley Publishing Co., Reading, MA, 1988.
12. I. J. Good, "The Interaction Algorithm and Practical Fourier Analysis," *J. Roy. Statist. Soc., Ser. B* **20**, 372–375 (1960).
13. U. Schumann and R. A. Sweet, "Fast Fourier Transforms for Direct Solution of Poisson's Equation with Staggered Boundary Conditions," *J. Comput. Phys.* **75**, No. 1, 123–137 (1988).

Received February 19, 1993; accepted for publication August 16, 1993

Bert L. Bradford *IBM Federal Systems Company, 6300 Diagonal Highway, Boulder, Colorado 80301 (BRADFORD at BLDFVM9, bradford@bldfvm9.vnet.ibm.com).* Dr. Bradford is an Advisory Engineer/Scientist in the Surveillance Algorithms Department at the Boulder facility. He holds a B.A. degree in mathematics and music from the University of North Texas, an M.A. degree in mathematics from the University of Texas at Austin, and a Ph.D. in applied mathematics from the University of Colorado. His doctoral program was supported by the IBM Resident Study Program. He joined IBM in 1979, and has worked in the areas of ground-based astrodynamics applications for the Space Shuttle as well as space-based infrared surveillance algorithms. Dr. Bradford is a member of the Society for Industrial and Applied Mathematics.