



3 4456 0353251 3

ORNL/TM-12032

# ornl

**OAK RIDGE  
NATIONAL  
LABORATORY**

**MARTIN MARIETTA**

## A Fortran 90 Code for Magnetohydrodynamics

### Part I: Banded Convolution

David W. Walker

OAK RIDGE NATIONAL LABORATORY  
 CENTRAL RESEARCH LIBRARY  
 CIRCULATION SECTION  
 450RN ROOM 175  
**LIBRARY LOAN COPY**  
 DO NOT TRANSFER TO ANOTHER PERSON  
 If you wish someone else to see this  
 report, send in name with report and  
 the library will arrange a loan.  
 (ORNL-788) (1-87)

MANAGED BY  
MARTIN MARIETTA ENERGY SYSTEMS, INC.  
FOR THE UNITED STATES  
DEPARTMENT OF ENERGY

This report has been reproduced directly from the best available copy.

Available to DOD and DOD contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831; prices available from (315) 576 8401, FTS 626 8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-12032

Engineering Physics and Mathematics Division

Mathematical Sciences Section

**A FORTRAN 90 CODE FOR MAGNETOHYDRODYNAMICS  
PART I: BANDED CONVOLUTION**

David W. Walker

Mathematical Sciences Section  
Oak Ridge National Laboratory  
P.O. Box 2008, Bldg. 6012  
Oak Ridge, TN 37831-6367

Date Published: March 1992

Research sponsored by the Office of Fusion Energy, U. S. Department of Energy.

Prepared by the  
Oak Ridge National Laboratory  
Oak Ridge, Tennessee 37831  
managed by  
Martin Marietta Energy Systems, Inc.  
for the  
U.S. DEPARTMENT OF ENERGY  
under Contract No. DE-AC05-84OR21400



MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES  
3 4456 0353251 3



## Contents

1	Introduction . . . . .	1
2	Banded Convolution . . . . .	1
3	Fortran 90 Implementation . . . . .	5
4	Banded Convolution on the CM-2 . . . . .	10
5	Conclusions . . . . .	15
6	Acknowledgements . . . . .	16
7	References . . . . .	17



**A FORTRAN 90 CODE FOR MAGNETOHYDRODYNAMICS  
PART I: BANDED CONVOLUTION**

David W. Walker

**Abstract**

This report describes progress in developing a Fortran 90 version of the KITE code for studying plasma instabilities in Tokamaks. In particular, the evaluation of convolution terms appearing in the numerical solution is discussed, and timing results are presented for runs performed on an 8k processor Connection Machine (CM-2). Estimates of the performance on a full-size 64k CM-2 are given, and range between 100 and 200 Mflops. The advantages of having a Fortran 90 version of the KITE code are stressed, and the future use of such a code on the newly announced CM5 and Paragon computers, from Thinking Machines Corporation and Intel, is considered.



## 1. Introduction

This is the first in a series of reports describing progress towards a Fortran 90 implementation of the KITE code, and similar spectral codes. The KITE code uses a magnetohydrodynamic (MHD) approach to study turbulence and transport in Tokamak plasmas, and has been used, for example, in investigations of tearing mode turbulence [4], and the dynamo effect in reversed-field pinch configurations [5]. A finite difference grid is used in the radial direction, and a Fourier series expansion in the poloidal and toroidal directions. Details of the equations and algorithms are given elsewhere ([3],[4]), and so will not be repeated here. The important point to note is that the two major computational tasks are,

- The evaluation of 2-D convolutions at each radial grid point. These arise from the Fourier representation of nonlinear terms in the governing PDEs.
- The solution of a block tridiagonal linear system for each mode included in the computation. These systems are due to implicit terms that arise in the radial discretization, and, in the models run so far, the size of the blocks ranges from 1 to 7, depending on the complexity of the model physics.

In this report we shall deal with the Fortran 90 implementation of the 2-D convolutions. Subsequent reports will consider the solution of the block tridiagonal systems, and the implementation of a complete Fortran 90 version of the KITE code.

## 2. Banded Convolution

For the types of problem being studied only  $(m, n)$  modes within a narrow helicity band are of interest, as shown in Fig. 1 in which the crosses indicate which modes are actually included in the computation. We shall refer to the convolutions performed as “banded convolutions”.

An alternative approach to implementing spectral codes like KITE is to regard the problem as a dense convolution and use fast Fourier transform methods. This method uses more memory than the banded convolution method, but requires less data movement, and may be the best approach on machines for which data movement is costly. In addition, direct convolution has a smaller operation count than the FFT method only for a sufficiently narrow band of modes. The FFT approach is being investigated by Kerbel [6] on the Connection Machine CM-2, for which optimized FFT routines already exist.

In the Fortran 77 version of the KITE code the modes of interest are labeled  $\ell = 1, 2, \dots, \ell_{\max}$ . Indirection arrays are initialized at the start of the KITE code to store which modal interactions contribute to a given mode. Thus the convolution of two arrays  $G$  and  $H$  for mode  $\ell = (m, n)$  is written in terms of index arrays  $K_g^\ell$  and  $K_h^\ell$  as follows,

$$F(\ell) = \sum_{\ell' \in \mathcal{P}(\ell)} S_\ell(\ell') \cdot G(K_g^\ell(\ell')) \cdot H(K_h^\ell(\ell')), \quad \text{for } \ell = 1, 2, \dots, \ell_{\max} \quad (1)$$

where  $\mathcal{P}(\ell)$  is the set of modes contributing to mode  $\ell$ , and  $S_\ell(\ell')$  is either -1, 0, or 1, depending on what types of function the arrays being convolved represent. For clarity we shall assume that  $S_\ell(\ell') = 1$  for the rest of this report.

Although Fortran 90 permits the use of vector subscripts to perform the indirect indexing in Eq. (1), the use of such constructs generally results in inefficient code on advanced architecture

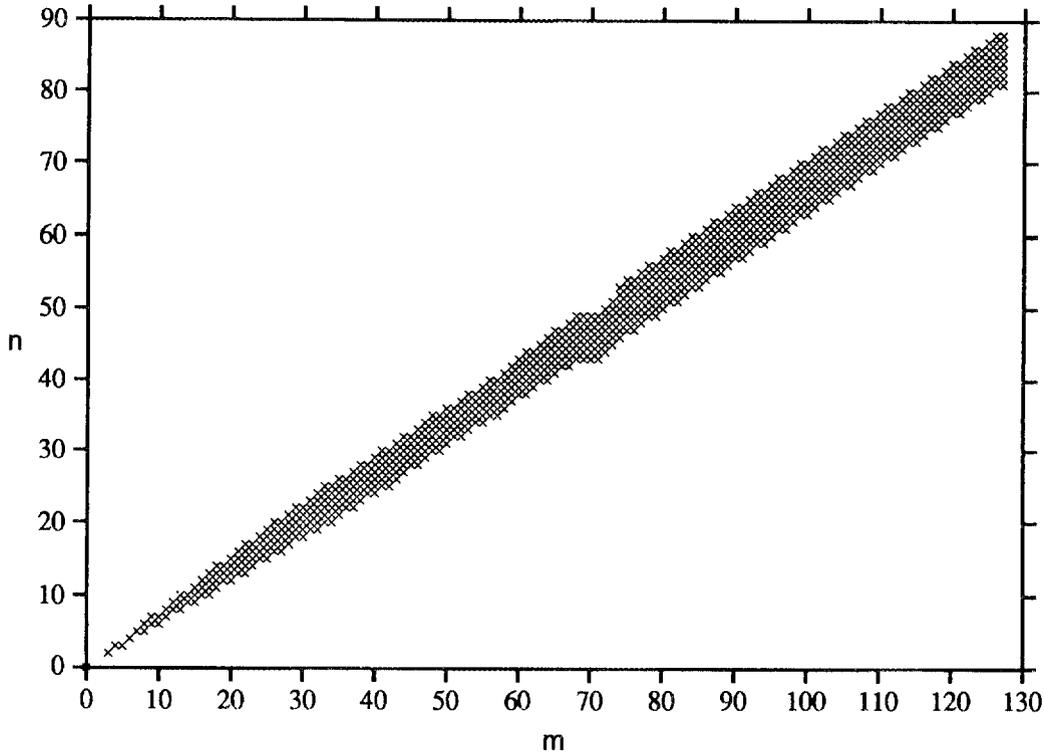


Figure 1.  $(m, n)$  modes are indicated by crosses.

machines, since it inhibits vectorization, results in a high cache miss ratio, and/or requires irregular interprocessor communication. In designing a Fortran 90 version of the KITE code we, therefore, avoid the use of indirect indexing. As we shall see, indirect indexing is “overkill” for the problem at hand since the problem does involve a certain regularity of structure. The use of indirect indexing only makes sense in genuinely sparse or irregular problems.

The banded convolutions that are evaluated in the KITE code at each radial grid point are actually of the form,

$$F(m, n) = \begin{cases} F_1(m, n) + F_2(m, n) & \text{for } (m, n) \neq (0, 0) \in \mathcal{S} \\ 0 & \text{for } (m, n) = (0, 0) \end{cases} \quad (2)$$

where  $\mathcal{S}$  is the set of modes used in the simulation, and,

$$\begin{aligned} F_1(m, n) &= \sum_{m'=m}^{M-1} \sum_{n'=n}^{n_2(m')} [G(m', n') \cdot H(m' - m, n' - n) + G(m' - m, n' - n) \cdot H(m', n')] \\ F_2(m, n) &= \sum_{m'=1}^{m-1} \sum_{n'=n_*(m')}^{n-1} [G(m - m', n - n') \cdot H(m', n') + G(m', n') \cdot H(m - m', n - n')] \end{aligned} \quad (3)$$

where  $n_*(m') = \max(1, n_1(m'))$ . In Eq. (3) we assume that the index  $n$  runs from  $n_1(m)$  to  $n_2(m)$  for each  $m$ , and that  $G(m, n) = H(m, n) = 0$  for  $(m, n)$  not in  $\mathcal{S}$ .

The Fortran 90 language is designed for the convenient and efficient manipulation of arrays

(see, for example, [1]). We, therefore, must reformulate the convolution in Eq. (3) in terms of elementwise operations between conformal arrays without any indirect indexing. With this aim in mind, we next rewrite  $F_1(m, n)$  as,

$$\begin{aligned} F_1(m, n) &= \sum_{m'=0}^{M-1} \sum_{n'=0}^{N_1-1} [G(m+m', n+n') \cdot H(m', n') + G(m', n') \cdot H(m+m', n+n')] \\ &= \sum_{m'=0}^{M-1} \sum_{n'=0}^{N_1-1} [G(m+m', n+n'+n_1(m')) \cdot H(m', n'+n_1(m')) + \\ &\quad G(m', n'+n_1(m')) \cdot H(m+m', n+n'+n_1(m'))] \end{aligned} \quad (4)$$

where,

$$N_1 = \max_m (n_2(m) - n_1(m)) + 1 \quad (5)$$

In Eq. (4),  $n_1(m')$  has been added to the  $n'$  index because  $G(m', n') = H(m', n') = 0$  for  $n' < n_1(m')$ . Reformulating the expression for  $F_2(m, n)$  we have,

$$\begin{aligned} F_2(m, n) &= \sum_{m'=1}^{m-1} \sum_{n'=n_1(m')}^{n-1} [G(m-m', n-n') \cdot H(m', n') + G(m', n') \cdot H(m-m', n-n')] \\ &\quad - \sum_{m' \in \mathcal{M}(m)} [G(m', 0) \cdot H(m-m', n) + G(m-m', n) \cdot H(m', 0)] \\ &= \sum_{m'=1}^{M-1} \sum_{n'=0}^{N_1-1} [G(m-m', n-n'-n_1(m')) \cdot H(m', n'+n_1(m')) + \\ &\quad G(m', n'+n_1(m')) \cdot H(m-m', n-n'-n_1(m'))] \\ &\quad - \sum_{n' \in \mathcal{N}(m, n)} [G(0, n') \cdot H(m, n-n') + G(m, n-n') \cdot H(0, n')] \\ &\quad - 2 \sum_{m' \in \mathcal{M}(m)} [G(m', 0) \cdot H(m-m', n) + G(m-m', n) \cdot H(m', 0)] \\ &\quad - G(0, 0) \cdot H(m, n) - G(m, n) \cdot H(0, 0) \end{aligned} \quad (6)$$

where we define the following index sets,

$$\begin{aligned} \mathcal{M}(m) &= \{m' \mid (m', 0) \in \mathcal{S} \text{ and } m' \in \langle 1, m-1 \rangle\} \\ \mathcal{N}(m, n) &= \{n' \mid (0, n') \in \mathcal{S} \text{ and } n' \in \langle 1, n-n_1(m) \rangle\} \end{aligned} \quad (7)$$

where  $\langle x, y \rangle$  denotes the set of integers in the range  $x$  to  $y$ , inclusive.

For dense convolution of  $M \times N$  arrays,  $n_1(m) = 0$  and  $N_1 = N$  so  $\mathcal{M}(m) = \langle 1, m-1 \rangle$ , and  $\mathcal{N}(m, n) = \langle 1, n \rangle$ . For the types of banded convolutions that are of interest here we shall assume that the only mode for which either  $m$  or  $n$  is zero is the  $(0, 0)$  mode. Under this assumption  $\mathcal{M}(m)$  and  $\mathcal{N}(m, n)$  are empty sets for  $m, n > 1$ . The distribution of modes in Fig. 1 is of this type. With this assumption we may write,

$$\begin{aligned} F_2(m, n) &= \sum_{m'=1}^{M-1} \sum_{n'=0}^{N_1-1} [G(m-m', n-n'-n_1(m')) \cdot H(m', n'+n_1(m')) + \\ &\quad G(m', n'+n_1(m')) \cdot H(m-m', n-n'-n_1(m'))] \end{aligned} \quad (8)$$

$$-G(0, 0) \cdot H(m, n) - G(m, n) \cdot H(0, 0)$$

Now we have transformed the banded convolution in Eq. (3) into a dense convolution in Eqs. (4) and (8), and we can further modify the convolution so that it involves only dense arrays as follows,

$$\begin{aligned} f_1(m, n) &= \sum_{m'=0}^{M-1} \sum_{n'=0}^{N_1-1} [g(m+m', n+n'+n_1(m)+n_1(m')-n_1(m+m')) \cdot h(m', n') + \\ &\quad g(m', n') \cdot h(m+m', n+n'+n_1(m)+n_1(m')-n_1(m+m'))] \\ f_2(m, n) &= \sum_{m'=1}^{M-1} \sum_{n'=0}^{N_1-1} [g(m-m', n-n'+n_1(m)-n_1(m')-n_1(m-m')) \cdot h(m', n') + \\ &\quad g(m', n') \cdot h(m-m', n-n'+n_1(m)-n_1(m')-n_1(m-m'))] \\ &\quad -g(0, 0) \cdot h(m, n) - g(m, n) \cdot h(0, 0) \end{aligned} \quad (9)$$

where,

$$g(m, n) = G(m, n + n_1(m)), \quad h(m, n) = H(m, n + n_1(m)) \quad (10)$$

with similar relationships existing between  $f_1$  and  $F_1$ , and between  $f_2$  and  $F_2$ . Finally, for notational convenience, we introduce two offsets,

$$\begin{aligned} k_1(m, m') &= n_1(m) + n_1(m') - n_1(m+m') \\ k_2(m, m') &= n_1(m) - n_1(m') - n_1(m-m') \end{aligned} \quad (11)$$

We also take into account the symmetry between  $g$  and  $h$  in the expressions for  $f_1$  and  $f_2$ , and write the convolution as,

$$\begin{aligned} f_1(m, n) &= f_1^{(g,h)}(m, n) + f_1^{(h,g)}(m, n) \\ f_2(m, n) &= f_2^{(g,h)}(m, n) + f_2^{(h,g)}(m, n) \end{aligned} \quad (12)$$

for  $m = 1, \dots, M-1$ , and  $n = 1, \dots, N_1-1$ , where,

$$f_1^{(x,y)}(m, n) = \sum_{m'=0}^{M-1} \sum_{n'=0}^{N_1-1} x(m+m', n+n'+k_1(m, m')) \cdot y(m', n') \quad (13)$$

$$\begin{aligned} f_2^{(x,y)}(m, n) &= \sum_{m'=1}^{M-1} \sum_{n'=0}^{N_1-1} x(m-m', n-n'+k_2(m, m')) \cdot y(m', n') \\ &\quad -x(0, 0) \cdot y(m, n) \end{aligned} \quad (14)$$

Eqs. (13) and (14) recast the banded convolution in a form that may be readily implemented in Fortran 90. It is also possible to express Eqs. (13) and (14) in terms of matrix-vector products, and this gives some insight into the Fortran 90 implementation. Thus, we write,

$$f_1^{(x,y)}(m, n) = \sum_{m'=0}^{M-1} Z_{m,m'}^+(n) \quad (15)$$

$$f_2^{(x,y)}(m, n) = \sum_{m'=1}^{M-1} Z_{m,m'}^-(n) - x(0, 0) \cdot y(m, n) \quad (16)$$

where,

$$Z_{m,m'}^+(n) = \sum_{n'=0}^{N_1-1} X_{m,m'}^+(n+n') \cdot Y_{m'}(n') \quad (17)$$

$$Z_{m,m'}^-(n) = \sum_{n'=0}^{N_1-1} X_{m,m'}^-(n-n') \cdot Y_{m'}(n') \quad (18)$$

and

$$\begin{aligned} X_{m,m'}^+(q) &= x(m+m', q+k_1(m, m')), & Y_{m'}(q) &= y(m', q) \\ X_{m,m'}^-(q) &= x(m-m', q+k_2(m, m')), \end{aligned} \quad (19)$$

With this notation  $Z_{m,m'}^+$  can be expressed as the product of a Hankel matrix, with zeros below the minor diagonal, and a vector,

$$\begin{bmatrix} Z^+(0) \\ Z^+(1) \\ \vdots \\ Z^+(N_1-1) \end{bmatrix}_{m,m'} = \begin{bmatrix} X^+(0) & X^+(1) & \dots & X^+(N_1-1) \\ X^+(1) & \dots & X^+(N_1-1) & 0 \\ \vdots & \ddots & \ddots & \vdots \\ X^+(N_1-1) & 0 & \dots & 0 \end{bmatrix}_{m,m'} \cdot \begin{bmatrix} Y(0) \\ Y(1) \\ \vdots \\ Y(N_1-1) \end{bmatrix}_{m'} \quad (20)$$

where for notational clarity the subscripts on  $Z^+$ ,  $X^+$ ,  $X^-$ , and  $Y$  have been used to label the matrices, rather than the matrix elements. Similarly,  $Z_{m,m'}^-$  can be expressed as the product of a lower-triangular Toeplitz matrix and a vector,

$$\begin{bmatrix} Z^-(0) \\ Z^-(1) \\ \vdots \\ Z^-(N_1-1) \end{bmatrix}_{m,m'} = \begin{bmatrix} X^-(0) & 0 & \dots & 0 \\ X^-(1) & \ddots & \ddots & \vdots \\ \vdots & \ddots & X^-(0) & 0 \\ X^-(N_1-1) & \dots & X^-(1) & X^-(0) \end{bmatrix}_{m,m'} \cdot \begin{bmatrix} Y(0) \\ Y(1) \\ \vdots \\ Y(N_1-1) \end{bmatrix}_{m'} \quad (21)$$

### 3. Fortran 90 Implementation

To clarify the basic structure of the Fortran 90 implementation of banded convolution we shall first consider the simple case in which  $n_1(m) = m$ . In this case  $k_1(m, m') = k_2(m, m') = 0$ , and we have,

$$f_1^{(x,y)}(m, n) = \sum_{m'=0}^{M-1} \sum_{n'=0}^{N_1-1} x(m+m', n+n') \cdot y(m', n') \quad (22)$$

$$f_2^{(x,y)}(m, n) = \sum_{m'=1}^{M-1} \sum_{n'=0}^{N_1-1} x(m-m', n-n') \cdot y(m', n') - x(0, 0) \cdot y(m, n) \quad (23)$$

From Eqs. (22) and (23) it clear that the Fortran 90 implementation of banded convolution has a doubly-nested loop structure. On each pass through the loop the elementwise product of two matrices must be accumulated in a third result matrix. On pass  $m', n'$  through the loop

one of the product matrices has all elements set to  $y(m', n')$ . In Fortran 90 such a matrix can be generated using the SPREAD function, which broadcasts copies of a source array along a specified dimension. In evaluating  $f_1^{(x,y)}$ , the  $(m, n)$ th element of the second product matrix is  $x(m + m', n + n')$ , which can be generated at each pass through the loop by shifting copies of the convolution array,  $x$ , using the Fortran 90 function EOSHIFT. To find  $f_2^{(x,y)}$  we need simply shift a copy of array  $x$  in the opposite direction. It should be noted that the use of the function EOSHIFT ensures that the appropriate elements in the shifted product array get set to zero, as required by the condition  $G(m, n) = H(m, n) = 0$  for  $(m, n)$  not in  $\mathcal{S}$ . Thus, the Fortran 90 implementation of a dense, 2-D convolution can be written as shown in Fig. 2. In an efficient implementation, the evaluation of  $f_1^{(x,y)}$  and  $f_2^{(x,y)}$  may be performed within the same double loop in order to reduce the number of calls to the SPREAD function. For clarity we have assumed in Fig. 2 that separate loops are used. `mnmask` is a logical array that indicates which modes are included in the simulation.

```
      :
      f1 = 0.0
      xsm = x
      DO MPRIME=0,M-1
      ysm = SPREAD (y(:,MPRIME,:), DIM=2, NCOPIES=M)
      xshift = xsm
      DO NPRIME=0,N1-1
      yspred = SPREAD (ysm(:, :, NPRIME), DIM=3, NCOPIES=N1)
      WHERE (mnmask)
      f1 = f1 + xshift*yspred
      END WHERE
      xshift = EOSHIFT (xshift, DIM=3, SHIFT=1)
      END DO
      xsm = EOSHIFT (xsm, DIM=2, SHIFT=1)
      END DO
      :
```

Figure 2(a). Fortran 90 code to find  $f_1^{(x,y)}$  for banded convolution when  $n_1(m) = m$ .

It should be noted that in Fig. 2 all arrays are three-dimensional since we must evaluate a convolution at each radial grid point. Thus, the first array dimension corresponds to the radial grid point index,  $j$ , and the second and third dimensions to the Fourier mode indices,  $m$  and  $n$ , respectively. So, for example, an array `x` might be declared as follows;

```
PARAMETER (JSTAR=128, MSTAR=128, NSTAR=16)
REAL, DIMENSION(0:JSTAR-1,0:MSTAR-1,0:NSTAR-1) :: x
```

```

      :
      f2 = 0.0
      xsm = EOSHIFT (x, DIM=2, SHIFT=-1)
      DO MPRIME=1,M-1
        ysm = SPREAD (y(:,MPRIME,:), DIM=2, NCOPIES=M)
        xshift = xsm
        DO NPRIME=0,N1-1
          yspred = SPREAD (ysm(:, :, NPRIME), DIM=3, NCOPIES=N1)
          WHERE (mnmask)
            f2 = f2 + xshift*yspred
          END WHERE
          xshift = EOSHIFT (xshift, DIM=3, SHIFT=-1)
        END DO
        xsm = EOSHIFT (xsm, DIM=2, SHIFT=-1)
      END DO
      WHERE (mnmask)
        f2 = f2 - y*SPREAD (SPREAD (x(:,0,0),DIM=2,NCOPIES=M),DIM=3,NCOPIES=N1)
      END WHERE
      :

```

Figure 2(b). Fortran 90 code to find  $f_2^{(x,y)}$  for banded convolution when  $n_1(m) = m$ .

The algorithm for banded convolution shown in Figs. 2(a) and (b) can also be interpreted in terms of the matrix-vector representation of Eqs. (20) and (21), if these two equations are rewritten as,

$$\begin{bmatrix} Z^+(0) \\ Z^+(1) \\ \vdots \\ Z^+(N_1 - 1) \end{bmatrix}_{m,m'} = Y_{m'}(0) \begin{bmatrix} X^+(0) \\ X^+(1) \\ \vdots \\ X^+(N_1 - 1) \end{bmatrix}_{m,m'} + Y_{m'}(1) \begin{bmatrix} X^+(1) \\ \vdots \\ X^+(N_1 - 1) \\ 0 \end{bmatrix}_{m,m'} \\
 + \dots + Y_{m'}(N_1 - 1) \begin{bmatrix} X^+(N_1 - 1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{m,m'} \quad (24)$$

and,

$$\begin{aligned}
 \begin{bmatrix} Z^-(0) \\ Z^-(1) \\ \vdots \\ Z^-(N_1-1) \end{bmatrix}_{m,m'} &= Y_{m'}(0) \begin{bmatrix} X^-(0) \\ X^-(1) \\ \vdots \\ X^-(N_1-1) \end{bmatrix}_{m,m'} + Y_{m'}(1) \begin{bmatrix} 0 \\ X^-(0) \\ \vdots \\ X^-(N_1-2) \end{bmatrix}_{m,m'} \\
 &+ \cdots + Y_{m'}(N_1-1) \begin{bmatrix} 0 \\ \vdots \\ 0 \\ X^-(0) \end{bmatrix}_{m,m'} \quad (25)
 \end{aligned}$$

Each pass through the inner loop in Fig. 2(a) or (b) evaluates and accumulates one term on the righthand side of Eq. (24) or (25). It is clear from Eqs. (24) and (25) how the call to EOSHIFT in the inner loop shifts the  $X_{m,m'}^+$  and  $X_{m,m'}^-$  vectors in the evaluation of the vectors  $Z_{m,m'}^+$  and  $Z_{m,m'}^-$ , respectively.

Performing the banded convolution as prescribed by Eqs. (12)–(14) when the offsets  $k_1$  and  $k_2$  are nonzero is somewhat more difficult because we need to shift the  $x$  array by the appropriate offset before entering the inner loop. Since the offsets may be positive or negative the Fortran 90 function CSHIFT must be used to shift the  $x$  array by the correct offset in the outer loop, and to rotate the array in the inner loop. CSHIFT performs these shifts periodically, and so ensures that we don't "lose" values that are shifted off the end of an array when applying an offset, but which must be rotated back into use in the inner loop. The use of CSHIFT places the following constraint on the dimensioning of arrays,

$$\begin{aligned}
 \max_{0 \leq m' < M-m} (n_2(m') - n_1(m') - k_1(m, m')) &< N_* \\
 \max_{1 \leq m' < m} (n_2(m') - n_1(m') + k_2(m, m')) &< N_* \quad (26)
 \end{aligned}$$

for all  $m$ , where  $N_*$  corresponds to **NSTAR** in the parameter statement above. A check must be made at the start of the program to ensure that these conditions are satisfied. Since many convolutions are performed in each time step, the overhead in performing this check will be amortized over the computation.

In order to evaluate the offsets  $k_1(m, m')$  and  $k_2(m, m')$  the SPREAD and EOSHIFT functions must be applied to  $n_1$ . The Fortran 90 code for performing banded convolution with nonzero offsets is shown in Figs. 3(a) and (b).

```
      :  
      f1 = 0.0  
      n1sh = n1  
      lshif = lmask  
      xsm = x  
      DO MPRIME=0,M-1  
        IF ( LDOIT(MPRIME) ) THEN  
          ysm = SPREAD (y(:,MPRIME,:), DIM=2, NCOPIES=M)  
          n1sp = SPREAD (n1(:,MPRIME,:), DIM=2, NCOPIES=M)  
          k1 = 0  
          WHERE (lshif)  
            k1 = n1+n1sp-n1sh  
          END WHERE  
          xshift = CSHIFT (xsm, DIM=3, SHIFT=k1)  
          DO NPRIME=0,N1-1  
            yspred = SPREAD (ysm(:, :, NPRIME), DIM=3, NCOPIES=N1)  
            WHERE (mnmask)  
              f1 = f1 + xshift*yspred  
            END WHERE  
            xshift = CSHIFT (xshift, DIM=3, SHIFT=1)  
          END DO  
        END IF  
        lshif = EOSHIFT (lshif, DIM=2, SHIFT=1)  
        n1sh = EOSHIFT (n1sh, DIM=2, SHIFT=1)  
        xsm = EOSHIFT (xsm, DIM=2, SHIFT=1)  
      END DO  
      :  
      :
```

Figure 3(a). Fortran 90 code to find  $f_1^{(x,y)}$  for banded convolution

In Figs. 3(a) and (b) the one-dimensional logical array LDOIT is set to false if there are no modes included in the simulation for a particular value of  $m$ , and is true otherwise. `mnmask` is a logical array that indicates which modes are included in the simulation. The logical array `lmask` is the array LDOIT spread over the radial grid point index. The arrays LDOIT and `lmask` are needed to handle cases in which no modes are included in the model for certain values of  $m$  between 0 and  $M - 1$ . To ensure that sensible values are assigned to the offsets, `k1` and `k2`, the array `lmask` must be shifted in the outer loop. The arrays `k1`, `k2`, `lmask`, `lshif`, `n1`, `n1sp`, and `n1sh` are all two-dimensional arrays that are dimensioned the same as the lower 2 dimensions of the three-dimensional arrays `x`, `y`, `f1`, `f2`, `ysm`, `xsm`, `xshif`, `yspred`, and `mnmask`.

```

      :
      f2 = 0.0
      lshif = EOSHIFT (lmask, DIM=2, SHIFT=-1)
      n1sh = EOSHIFT (n1, DIM=2, SHIFT=-1)
      xsm = EOSHIFT (x, DIM=2, SHIFT=-1)
      DO MPRIME=1,M-1
        IF ( LDCIT(MPRIME) ) THEN
          ysm = SPREAD (y(:,MPRIME,:), DIM=2, NCOPIES=M)
          n1sp = SPREAD (n1(:,MPRIME,:), DIM=2, NCOPIES=M)
          k2 = 0
          WHERE (lshif)
            k2 = n1-n1sp-n1sh
          END WHERE
          xshift = CSHIFT (xsm, DIM=3, SHIFT=k2)
          DO NPRIME=0,N1-1
            yspred = SPREAD (ysm(:, :, NPRIME), DIM=3, NCOPIES=N1)
            WHERE (mnmask)
              f2 = f2 + xshift*yspred
            END WHERE
            xshift = CSHIFT (xshift, DIM=3, SHIFT=-1)
          END DO
        END IF
        lshif = EOSHIFT (lshif, DIM=2, SHIFT=-1)
        n1sh = EOSHIFT (n1sh, DIM=2, SHIFT=-1)
        xsm = EOSHIFT (xsm, DIM=2, SHIFT=-1)
      END DO
      WHERE (mnmask)
        f2 = f2 - y*SPREAD (SPREAD (x(:,0,0),DIM=2,NCOPIES=M),DIM=3,NCOPIES=N1)
      END WHERE
      :

```

Figure 3(b). Fortran 90 code to find  $f_2^{(x,y)}$  for banded convolution

#### 4. Banded Convolution on the CM-2

When implementing the Fortran 90 code for banded convolution on the Connection Machine CM-2 additional statements are used to indicate if and how arrays are to be distributed over the processing elements. By default, each element of a distributed matrix is assigned to a separate virtual processor (VP). However, by declaring one of the array dimensions to be a serial dimension we can assign a vector of elements to each VP. For example, the directive,

```
CMF$ LAYOUT x(.,.:SERIAL)
```

indicates that the array  $x$  is decomposed over just its first two dimensions, so the VP set is a 2-D

array, with the VP at location  $(j, m)$  containing the vector  $x(j, m, n)$  for  $n = 0, 1, \dots, N_* - 1$ . These directives are interpreted by the CM Fortran compiler, but are regarded as comment lines by other compilers, so the Connection Machine code can be compiled and run on any other machine with a Fortran 90 compiler with no modifications to the code. More complete details of the layout of distributed arrays on the CM-2 are given in the *CM Fortran Reference Manual* [8]. In Figs. 2 and 3, the variables in lower case reside on the Connection Machine, while upper case variables reside on the front end computer.

The Fortran 90 code for banded convolution described in Sec. 3 was run on a CM-2 with 8192 processors for a number of different problem sizes. The problems considered were based on the  $(m, n)$  modes shown in Fig. 1, with the size of the problem being characterized by  $M$ , the maximum value of  $m$  included in the computation. Values of  $M$  ranging from 70 to 127 were considered, corresponding to 333 to 783 modes. The number of radial grid points was held constant at 98, and the maximum number of modes for any given  $m$  was  $N_1 = 8$ . The 3-D arrays were dimensioned as in Sec. 3, i.e., as  $128 \times 128 \times 16$  arrays. Two different data layouts were investigated. In the first the data are distributed over all three dimensions, so the VP set is three-dimensional with each VP containing a single matrix element from each distributed 3-D matrix. In the second data layout considered the third dimension, corresponding to the  $n$  index, is declared to be a serial dimension, as in the example LAYOUT directive above. Thus, the data are distributed over just the radial grid point and  $m$  indices, and the VP set has a two-dimensional structure. The code in all cases was double precision (64 bits), and compiled with optimization turned on by means of the `-O` compiler flag.

In order to optimize the banded convolution code an execution profile was obtained for the  $M = 127$  case (783 modes) by inserting calls to the CMF Fortran timing routines in the code. Results for the code shown in Figs. 3(a) and (b), which will be referred to as "Version 1" of the code, are given in the columns headed "V1-2D" and "V1-3D" of Table 1. The two cases correspond to distributing the data over two and three dimensions, respectively.

Loop	Task	V1-2D	V1-3D	V2-2D	V2-3D
Outer	SPREAD	3.04	1.98	3.04	1.98
	CSHIFT	14.58	34.03	6.36	11.16
	EOSHIFT	3.94	2.80	3.85	2.72
Inner	SPREAD	3.68	14.71	3.68	14.79
	Evaluate $f_1, f_2$	4.56	4.56	4.56	4.56
	CSHIFT	3.05	16.96	3.05	16.97

Table 1. CM times in seconds for major sections of the banded convolution code

The V1-2D and V1-3D execution profiles show that an appreciable amount of time is spent on the CSHIFT calls in the outer loop, namely,

```
xshift = CSHIFT (xsm, DIM=3, SHIFT=k1)
```

and,

```
xshift = CSHIFT (xsm, DIM=3, SHIFT=k2)
```

In the V1-2D code these calls account for over 40% of the execution time. These CSHIFT calls are expensive because the SHIFT argument is an array, and so for each value of  $m$  the `xsm` array must be shifted by a different amount. Presumably each time this type of CSHIFT

is called the maximum and minimum offsets in the array specifying the shift ( $k_1$  or  $k_2$ ) must be computed so the appropriate number of shifts can be performed. In a production run of the KITE code many calls to the convolution routine are made, and in each call the following four quantities remain unchanged;

$$\begin{aligned} k_1^L(m') &= \min_m (k_1(m, m')), & k_1^U(m') &= \max_m (k_1(m, m')) \\ k_2^L(m') &= \min_m (k_2(m, m')), & k_2^U(m') &= \max_m (k_2(m, m')) \end{aligned} \quad (27)$$

so some of the overhead in the outer loop calls to CSHIFT can be avoided by precomputing these quantities and storing them in one-dimensional arrays. Then, when evaluating  $f_1^{(x,y)}$  in the outer loop, the correct offset can be applied to the  $x$  array by first shifting it by  $k_1^U(m')$  in one direction, and then shifting it by  $k_1^L(m')$  in the other direction, using a WHERE construct to assign the appropriate value to the array `xshift`. In Version 2 of the code, when computing  $f_1^{(x,y)}$ , we replace the evaluation of  $k_1$  and the call to CSHIFT in the outer loop by the code section shown in Fig. 4. In the evaluation of  $f_2^{(x,y)}$  a very similar code is used.

```

      :
      k1      = n1 + n1sp - n1sp
      k1sp    = SPREAD (k1, DIM=3, NCOPIES=NSTAR)
      xshift  = xsm
      DO K=1,K1U(MPRIME)
        WHERE (k1sp>0)
          xshift = CSHIFT (xshift, DIM=3, SHIFT=1)
          k1sp   = k1sp - 1
        END WHERE
      END DO
      DO K=1,-K1L(MPRIME)
        WHERE (k1sp<0)
          xshift = CSHIFT (xshift, DIM=3, SHIFT=-1)
          k1sp   = k1sp + 1
        END WHERE
      END DO
      :

```

Figure 4. Fortran 90 code to find `xshift` in the outer loop when evaluating  $f_1^{(x,y)}$  in Version 2 of the code

Table 1 also gives execution profiles for Version 2 of the code. Comparing Versions 1 and 2 it is clear that by introducing the modification shown in Fig. 4 to eliminate overhead in the outer loop calls to CSHIFT a significant improvement in performance is achieved. However, even in the best case (V2-2D) about 80% of the time is spent in moving data by calls to spread and shift routines. Table 1 also shows that distributing the data over two dimensions is faster than over three dimensions. This is because calls that spread data over the  $n$  index dimension,

or shift data in this direction, require communication between VPs in the 3-D case, whereas no communication is necessary in the 2-D case. It is interesting to note that the calls to SPREAD and EOSHIFT in the outer loop, which move data along the  $m$  index dimension, actually run a little faster in the 3-D case than in the 2-D case. This because the VP ratio is higher in the 3-D case, and so data must be communicated between fewer physical processors than in the 2-D case. Although the 2-D data decomposition runs faster on an 8k-processor CM-2 for the problem considered, the 3-D data decomposition should win out on larger machines as it allows more parallelism to be exploited. There is little point in running the code on more than 16k processors with a 2-D data decomposition, as the VP ratio will be less than 1 and some PEs will be idle. However, for production runs of the KITE code much larger problems will be considered, so a 2-D decomposition may still be fastest, even on a 64k-processor CM-2.

In Table 2 timings of the banded convolution are given for a number of different problem sizes. The modes included in the computation are as shown in Fig. 1, with problem size being determined by the value of  $M$ , the cutoff in the  $m$  index.  $S_{77}$  is the number of floating point operations per radial grid point required to convolve two arrays in the original Fortran 77 version of the KITE code.

Problem Size			Time in seconds (CM/elapsed)			
M	Modes	$S_{77}$	V1-2D	V1-3D	V2-2D	V2-3D
70	333	160524	17.22/17.22	41.36/41.36	11.67/11.68	31.16/31.16
80	407	243732	19.61/19.61	47.96/47.96	13.72/13.72	37.23/37.23
90	487	354204	22.09/22.09	54.22/54.22	15.76/15.76	42.50/42.50
100	567	489912	24.56/24.56	60.49/60.49	17.71/17.71	47.61/47.61
110	647	652440	27.04/27.04	66.61/66.61	19.75/19.75	52.90/52.99
120	727	842088	29.51/29.51	71.90/71.90	21.99/21.99	58.52/58.53
127	783	991080	31.25/31.25	73.48/73.49	23.43/23.43	62.24/62.24

Table 2. Timing results on an 8k CM-2.

A measure of how effectively the CM-2 is being used is given by the ratio of the number of floating point operations performed by the sequential code and the CM-2 code. This will be referred to as the "utilization ratio",  $U$ , and is given by,

$$U = \left(\frac{J}{J_*}\right) \left(\frac{M}{M_*}\right) \left(\frac{N_1}{N_*}\right) \left(\frac{S_{77}}{S_{90}}\right) \quad (28)$$

where  $J_* \times M_* \times N_*$  is the declared size of the 3-D data arrays, and  $S_{90} = 3M^2N_1^2$  is the number of floating point operations per radial grid point for the Fortran 90 code. The factor of 3 arises in the expression for  $S_{90}$  because  $f_1^{(x,y)}$  and  $f_2^{(x,y)}$  are evaluated within the same double loop, which saves one multiplication. If separate loops were used the factor would be 4. In all the timing runs  $J_* = 128$ ,  $M_* = 128$ , and  $N_* = 16$ . A small value of  $U$  indicates that the compute power of the CM-2 is not being used effectively. In Fig. 2 we plot the utilization rate and speed in Mflops for the problem sizes considered in Table 2 for the V2-2D version of the code. The Mflops rates,  $M_{77}$  and  $M_{90}$ , are computed using the floating point operation count of the Fortran 77 and Fortran 90 codes, respectively.  $M_{77}$  can be used to compare the performance of the Fortran 90 code on a computer such as the CM-2, with that of the original Fortran 77 code

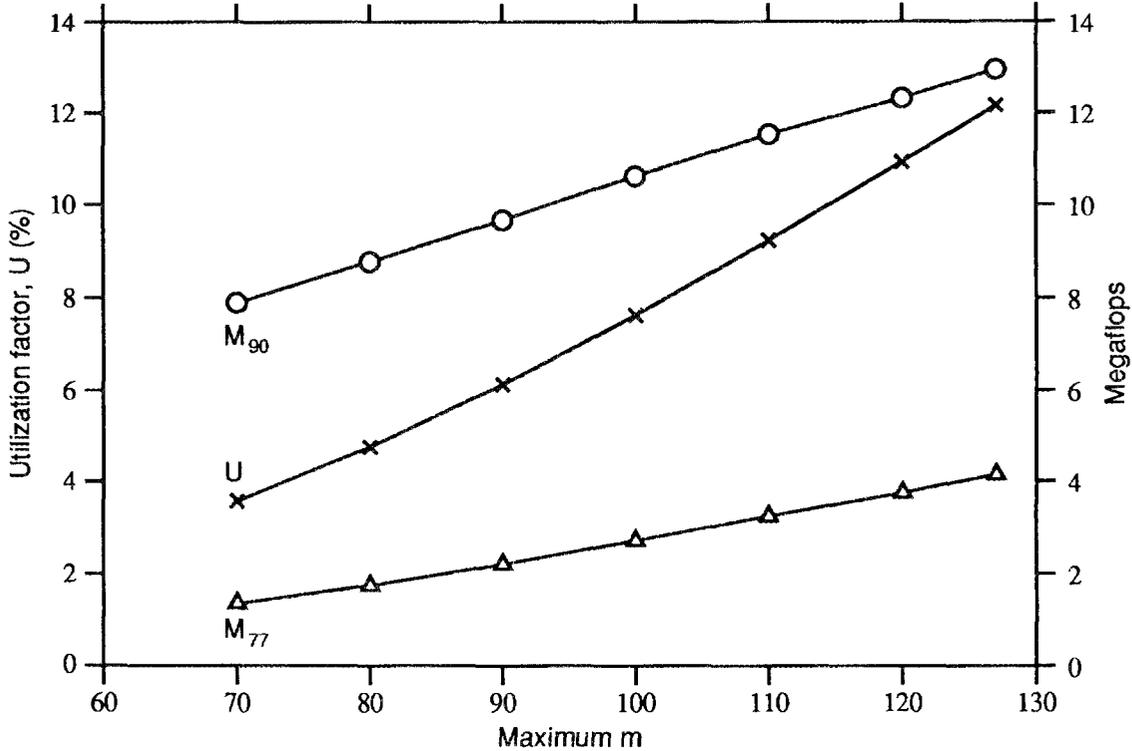


Figure 2. Utilization ratio and Mflops as a function of problem size for V2-2D

on some other machine, such as a workstation or vector supercomputer. The ratio of  $M_{77}$  for runs using the Fortran 90 and Fortran 77 codes equals the ratio of their respective execution times.  $M_{90}$  gives the actual execution rate in Megaflops for the Fortran 90 code.

For the problems considered the utilization ratio is always below 12.5%, so that the performance of the CM-2, as indicated by the Mflops rate, is rather poor, reaching only about  $M_{77} = 4$  and  $M_{90} = 13$  Mflops for the largest problem considered. The low value of  $U$  is due to the mismatch between the set of radial grid points and modes actually involved in the computation, and the 3-D arrays used to store these quantities. In Fig. 3 we plot  $n - n_1(m)$  against  $m$  to indicate which modes are involved in the  $M = 128$  problem. In this plot the blank area represents those array elements not actively involved in the computation at a particular radial grid point, and thus gives a measure of how much of the CM-2 is being wasted. The fact that  $N_1/N_* = 0.5$  means that half the compute power of the CM-2 is wasted because we are constrained to choose  $N_* = 16$  even though  $N_1 = 8$ . This constraint comes from the requirement that Eq. (26) be satisfied, and the fact that array dimensions on the CM-2 must be an exact power of two. A more carefully designed algorithm may eliminate this constraint, thereby halving the CM-2 execution times.

The expected performance in Mflops of the V2-2D code for a problem that fully utilizes the CM-2 may be estimated as,

$$M_{\max} = \frac{M_{77}}{U} = \frac{M_{90}}{(J/J_*)(M/M_*)(N_1/N_*)} \quad (29)$$

so extrapolating from the  $M = 128$  case we get a value of  $M_{\max} \approx 34$  Mflops for an 8k CM-2. This drops to 17 Mflops if the  $N_1/N_* = 0.5$  constraint applies. Scaling this up to a 64k CM-2,

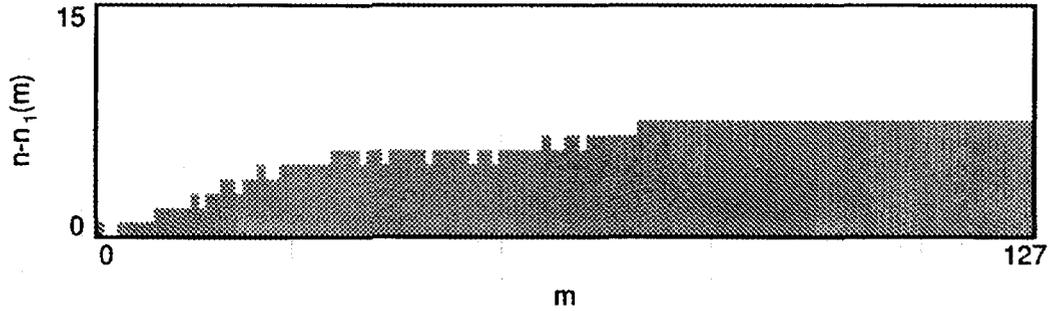


Figure 3. Schematic representation of active memory locations at a radial grid point.

we would expect a performance of about 136 Mflops for a problem that fully utilizes the CM-2, or 272 Mflops if the  $N_1/N_* = 0.5$  constraint can be eliminated without degrading performance.

It is worthwhile at this point to return to the question of whether direct convolution is faster than convolution using FFTs. If the evaluation of each FFT butterfly requires 10 floating point operations, and a real-to-complex FFT is used for the two forward and one inverse transforms required, then the operation count per radial grid point is approximately,

$$S_{\text{FFT}} = 3 \cdot 10 \cdot \left(\frac{MN}{4}\right) \log_2 \left(\frac{MN}{4}\right) \quad (30)$$

If  $M$  and  $N$  are required to be powers of 2 in the FFT version, i.e.,  $M = N = M_*$ , then the FFT method will have a lower operation count than the direct method if,

$$\log_2 \left(\frac{M_*}{2}\right) < 0.2N_1^2 \left(\frac{M}{M_*}\right)^2 \quad (31)$$

Thus, if the width of the band is  $N_1 = 4$ , and  $M = M_*$ , the direct method has a lower operation count if  $M_* \geq 32$ . However, if  $N_1$  is 8 the direct method only wins if  $M_* \geq 16384$ . Equation (31) shows how the choice of best algorithm depends sensitively on the width of the band, with the direct convolution method being appropriate only for very narrow bands of modes.

## 5. Conclusions

In comparing the performance of the Fortran 90 banded convolution code with the original Fortran 77 version the following factors must be considered;

1. The movement of data by calls to the SPREAD, CSHIFT, and EOSHIFT functions in the Fortran 90 code.
2. The ratio of the number of floating point operations per radial grid point for the Fortran 77 and Fortran 90 codes. The original Fortran 77 code uses indirect indexing, which results in a lower operation count since the Fortran 90 code is constrained to compute some modal interactions that are not required.
3. On some machines, such as the CM-2, grain size constraints may require array dimensions to be powers of two in the Fortran 90 code.

4. The ability of the codes to take advantage of advanced architectural features, such as vector and pipeline units, caching, and massive parallelism.

On multiprocessor systems the spreading and shifting of data may, or may not, result in inter-processor communication, and the efficiency with which these tasks are performed depends on the hardware, and the ability of the compiler to exploit it. On the CM-2 we have found it best to decompose the data over just the radial grid point and  $m$  indices, and on the commercial supercomputers currently available it is probably best to use as large a grain size as possible, subject to the requirement that all processing units have at least some data on which to work. This choice reflects the fact that data movement is expensive.

The direct convolution method has a lower operation count than the fast Fourier transform approach only for a sufficiently narrow band. However, the FFT convolution method requires more memory, and this may limit its usefulness on some machines.

On the CM-2 the requirement that array dimensions be powers of two significantly degrades performance for the problems considered. This problem is exacerbated by the constraint on  $N_*$  imposed by Eq. (26). Future work will look at efficient ways of removing this constraint, and further tuning the code for the CM-2.

A major advantage of the Fortran 90 code is its ability to make efficient use of the advanced architectural features of modern supercomputers. The original Fortran 77 code made use of indirect indexing to reduce the operation count and memory usage, however, this also inhibits vectorization and results in the inefficient use of cache. In the original code the loop over radial grid points was made the inner loop in order to increase the vector length, and improve caching. In a recent MIMD Fortran 77 implementation of the KITE code ([2],[7]), the code was parallelized by decomposing the data over just the radial grid point index. This approach allowed the code to be ported to machines such as the Intel iPSC/860 hypercube with only few modifications. In particular the indirect indexing was retained in the MIMD code. However, since the data are distributed over the radial grid point index the vector length is reduced, so that the pipeline units of the i860 cannot be exploited efficiently. The cache hit ratio is also low. The Fortran 90 code would avoid these problems.

A second important advantage of the Fortran 90 code is its portability. To port the Fortran 90 banded convolution code to a new machine one just needs to specify how the data are decomposed. On the CM-2 this is done by means of a LAYOUT directive. In the near future we expect Fortran 90 compilers to become available on all supercomputers, and their ability to exploit these machines to steadily improve.

Although the performance of the banded convolution on the CM-2 was rather disappointing for the problems considered, we believe it is important to develop a Fortran 90 version of the KITE code, with a view to implementing it on the next generation of concurrent supercomputers. Potential target machines include Thinking Machines Corporation's CM-5, Intel's Paragon, and new machines from Kendall Square and Alliant. The portability of Fortran 90 codes, and their ability to exploit advanced architectural features, justify the effort required to convert the KITE code from a Fortran 77 code using indirect indexing to an array-oriented Fortran 90 code.

## 6. Acknowledgements

The results reported in this work were obtained on the Connection Machine CM-2X at Sandia National Laboratories, Albuquerque, NM.

## 7. References

- [1] W. S. Brainerd, C. H. Goldberg, and J. C. Adams. *Programmers Guide to Fortran 90*. McGraw-Hill, 1990.
- [2] B. A. Carreras, N. Dominguez, J. B. Drake, J. N. Leboeuf, L. A. Charlton, J. A. Holmes, D. K. Lee, V. E. Lynch, and L. Garcia. Plasma turbulence calculations on supercomputers. *Int. J. Supercomputer Applications*, 4:97-110, 1990.
- [3] L. Garcia, H. R. Hicks, B. A. Carreras, L. A. Charlton, and J. A. Holmes. 3d nonlinear mhd calculations using implicit and explicit time integration schemes. *J. Comput. Phys*, 65:253, 1986.
- [4] H. R. Hicks, B. A. Carreras, J. A. Holmes, D. K. Lee, and B. V. Waddell. 3d nonlinear calculations of resistive tearing modes. *J. Comput. Phys.*, 44:46-69, 1981.
- [5] J. A. Holmes, B. A. Carreras, P. H. Diamond, and V. E. Lynch. Nonlinear dynamics of tearing modes in the reversed field pinch. *Physics of Fluids*, 31:1166, 1988.
- [6] G. Kerbel. Private communication, Summer 1991.
- [7] V. E. Lynch, B. A. Carreras, J. B. Drake, and J. N. Leboeuf. Plasma turbulence calculations on the Intel iPSC/860 (RX) hypercube. *Computing Systems in Engineering*, 2:299-305, 1991.
- [8] Thinking Machines Corporation, 245 First Street, Cambridge, MA 02142-1264. *CM Fortran Reference Manual*, July 1991.



**INTERNAL DISTRIBUTION**

- |        |                |        |                                 |
|--------|----------------|--------|---------------------------------|
| 1.     | B. R. Appleton | 19-23. | R. F. Sincovec                  |
| 2.     | C. Bottcher    | 24.    | G. M. Stocks                    |
| 3.     | B. A. Carreras | 25.    | M. R. Strayer                   |
| 4-5.   | T. S. Darland  | 26-30. | D. W. Walker                    |
| 6.     | E. D'Azevedo   | 31-35. | R. C. Ward                      |
| 7.     | J. J. Dongarra | 36.    | P. H. Worley                    |
| 8.     | J. B. Drake    | 37.    | Central Research Library        |
| 9.     | T. H. Dunigan  | 38.    | ORNL Patent Office              |
| 10.    | R. E. Flanery  | 39.    | K-25 Applied Technology Library |
| 11.    | J. N. Leboeuf  | 40.    | Y-12 Technical Library          |
| 12.    | V. E. Lynch    | 41.    | Laboratory Records - RC         |
| 13.    | C. E. Oliver   | 42-43. | Laboratory Records Department   |
| 14-18. | S. A. Raby     |        |                                 |

**EXTERNAL DISTRIBUTION**

44. Christopher R. Anderson, Department of Mathematics, University of California, Los Angeles, CA 90024
45. David C. Bader, Atmospheric and Climate Research Division, Office of Health and Environmental Research, Office of Energy Research, ER-76, U.S. Department of Energy, Washington, DC 20585
46. David H. Bailey, NASA Ames, Mail Stop 258-5, NASA Ames Research Center, Moffet Field, CA 94035
47. Edward H. Barsis, Computer Science and Mathematics, P. O. Box 5800, Sandia National Laboratory, Albuquerque, NM 87185
48. Colin Bennett, Department of Mathematics, University of South Carolina, Columbia, SC 29208
49. Dominique Bennett, CERFACS, 42 Avenue Gustave Coriolis, 31057 Toulouse Cedex, FRANCE
50. Marsha J. Berger, Courant Institute of Mathematical Sciences, 251 Mercer Street, New York, NY 10012
51. Mike Berry, Department of Computer Science, University of Tennessee, 107 Ayres Hall, Knoxville, TN 37996-1301
52. Ake Bjorck, Department of Mathematics, Linkoping University, S-581 83 Linkoping, Sweden

53. John H. Bolstad, Lawrence Livermore National Laboratory, L-16, P. O. Box 808, Livermore, CA 94550
54. George Bourianoff, Superconducting Super Collider Laboratory, 2550 Beckleymeade Avenue, Suite 260, Dallas, TX 75237-3946
55. Ralph G. Brickner, Los Alamos National Laboratory, Mail Stop B265, C-3, Los Alamos, NM 87545
56. Roger W. Brockett, Wang Professor of EE and CS, Division of Applied Sciences, Harvard University, Cambridge, MA 02138
57. Bill L. Buzbee, National Center for Atmospheric Research, P. O. Box 3000, Boulder, CO 80307
58. Captain Edward A. Carmona, Parallel Computing Research Group, Phillips Laboratory, Kirtland AFB, Albuquerque, NM 87117
59. John Cavallini, Acting Director, Scientific Computing Staff, Applied Mathematical Sciences, Office of Energy Research, U.S. Department of Energy, Washington, DC 20585
60. I-liang Chern, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439
61. Ray Cline, Sandia National Laboratories, Livermore, CA 94550
62. Alexandre Chorin, Mathematics Department, Lawrence Berkeley Laboratory, Berkeley, CA 94720
63. James Coronas, Ames Laboratory, Iowa State University, Ames, IA 50011
64. Jean Coté, RPN, 2121 Transcanada Highway, Suite 508, Dorval, Quebec H9P 1J3, CANADA
65. John J. Dorning, Department of Nuclear Engineering Physics, Thornton Hall, McCormick Road, University of Virginia, Charlottesville, VA 22901
66. Larry Dowdy, Computer Science Department, Vanderbilt University, Nashville, TN 37235
67. Iain S. Duff, Atlas Centre, Rutherford Appleton Laboratory, Didcot, Oxon OX11 0QX, England
68. John Dukowicz, Los Alamos National Laboratory, Group T-3, Los Alamos, NM 87545
69. Richard E. Ewing, Department of Mathematics, University of Wyoming, Laramie, WY 82071
70. Ian Foster, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439
71. Geoffrey C. Fox, NPAC, 111 College Place, Syracuse University, Syracuse, NY 13244-4100
72. Chris Fraley, Statistical Sciences, Inc., 1700 Westlake Ave. N, Suite 500, Seattle, WA 98119

73. Paul O. Frederickson, RIACS, MS 230-5, NASA Ames Research Center, Moffet Field, CA 94035
74. Dennis B. Gannon, Computer Science Department, Indiana University, Bloomington, IN 47401
75. J. Alan George, Vice President, Academic and Provost, Needles Hall, University of Waterloo, Waterloo, Ontario, CANADA N2L 3G1
76. James Glimm, Department of Mathematics, State University of New York, Stony Brook, NY 11794
77. Gene Golub, Computer Science Department, Stanford University, Stanford, CA 94305
78. John Gustafson, 236 Wilhelm, Ames Laboratory, Iowa State University, Ames, IA 50011
79. Phil Gresho, Lawrence Livermore National Laboratory, L-262, P. O. Box 808, Livermore, CA 94550
80. William D. Gropp, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439
81. Eric Grosse, AT&T Bell Labs 2T-504, Murray Hill, NJ 07974
82. James J. Hack, National Center for Atmospheric Research, P. O. Box 3000, Boulder, CO 80307
83. Michael T. Heath, NCSA, University of Illinois, 4157 Beckman Institute, 405 North Mathews Avenue, Urbana, IL 61801-2300
84. Michael Henderson, Los Alamos National Laboratory, Group T-3, Los Alamos, NM 87545
85. Lennart Johnsson, Thinking Machines Inc., 245 First Street, Cambridge, MA 02142-1214
86. Kirk Jordan Thinking Machines Inc., 245 First Street, Cambridge, MA 02142-1214
87. Malvyn Kalos, Cornell Theory Center, Engineering and Theory Center Bldg., Cornell University, Ithaca, NY 14853-3901
88. Hans Kaper, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439
89. Alan H. Karp, IBM Scientific Center, 1530 Page Mill Road, Palo Alto, CA 94304
90. Kenneth Kennedy, Department of Computer Science, Rice University, P. O. Box 1892, Houston, Texas 77001
91. Gary D. Kerbel Lawrence Livermore National Laboratory Mail Stop L-561 7000 East Avenue Livermore, CA 94550
92. Tom Kitchens, ER-7, Applied Mathematical Sciences, Scientific Computing Staff, Office of Energy Research, Office G-437 Germantown, Washington, DC 20585
93. Peter D. Lax, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012

94. James E. Leiss, Rt. 2, Box 142C, Broadway, VA 22815
95. Rich Loft, National Center for Atmospheric Research, P. O. Box 3000, Boulder, CO 80307
96. Michael C. MacCracken, Lawrence Livermore National Laboratory, L-262, P. O. Box 808, Livermore, CA 94550
97. Robert Malone, Los Alamos National Laboratory, C-3, Mail Stop B265, Los Alamos, NM 87545
98. Len Margolin, Los Alamos National Laboratory, Los Alamos, NM 87545
99. Frank McCabe, Department of Computing, Imperial College of Science and Technology, 180 Queens Gate, London SW7 2BZ, ENGLAND
100. James McGraw, Lawrence Livermore National Laboratory, L-306, P. O. Box 808, Livermore, CA 94550
101. Paul C. Messina, Mail Code 158-79, California Institute of Technology, 1201 E. California Blvd. Pasadena, CA 91125
102. Neville Moray, Department of Mechanical and Industrial Engineering, University of Illinois, 1206 West Green Street, Urbana, IL 61801
103. V. E. Oberacker, Department of Physics, Vanderbilt University, Box 1807, Station B, Nashville, TN 37235
104. Joseph Olinger, Computer Science Department, Stanford University, Stanford, CA 94305
105. Robert O'Malley, Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180-3590
106. James M. Ortega, Department of Applied Mathematics, Thornton Hall, University of Virginia, Charlottesville, VA 22901
107. Ron Peierls, Applied Mathematical Department, Brookhaven National Laboratory, Upton, NY 11973
108. Richard Pelz, Dept. of Mechanical and Aerospace Engineering, Rutgers University, Piscataway, NJ 08855-0909
109. Paul Pierce, Intel Scientific Computers, 15201 N.W. Greenbrier Parkway, Beaverton, OR 97006
110. Robert J. Plemmons, Departments of Mathematics and Computer Science, North Carolina State University, Raleigh, NC 27650
111. Jesse Poore, Computer Science Department, University of Tennessee, Knoxville, TN 37996-1300
112. Andrew Priestley, Institute for Computational Fluid Dynamics, Reading University, Reading RG6 2AX, ENGLAND
113. Daniel A. Reed, Computer Science Department, University of Illinois, Urbana, IL 61801

114. Lee Riedinger, Director, The Science Alliance Program, University of Tennessee, Knoxville, TN 37996
115. Garry Rodrigue, Numerical Mathematics Group, Lawrence Livermore National Laboratory, Livermore, CA 94550
116. Ahmed Sameh, Center for Supercomputing R & D, 1384 W. Springfield Avenue, University of Illinois, Urbana, IL 61801
117. Dave Schneider University of Illinois at Urbana-Champaign, Center for Supercomputing Research and Development, 319E Talbot - 104 S. Wright Street Urbana, IL 61801
118. David S. Scott, Intel Scientific Computers, 15201 N.W. Greenbrier Parkway, Beaverton, OR 97006
119. Robert Schreiber, RIACS, MS 230-5, NASA Ames Research Center, Moffet Field, CA 94035
120. William C. Skamarock, 3973 Escuela Court, Boulder, CO 80301
121. Richard Smith, Los Alamos National Laboratory, Group T-3, Mail Stop B2316, Los Alamos, NM 87545
122. Peter Smolarkiewicz, National Center for Atmospheric Research, MMM Group, P. O. Box 3000, Boulder, CO 80307
123. Jurgen Steppeler, DWD, Frankfurterstr 135, 6050 Offenbach, WEST GERMANY
124. Rick Stevens, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439
125. Paul N. Swarztrauber, National Center for Atmospheric Research, P. O. Box 3000, Boulder, CO 80307
126. Wei Pai Tang, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
127. Harold Trease, Los Alamos National Laboratory, Mail Stop B257, Los Alamos, NM 87545
128. Robert G. Voigt, ICASE, MS 132-C, NASA Langley Research Center, Hampton, VA 23665
129. Mary F. Wheeler, Rice University, Department of Mathematical Sciences, P. O. Box 1892, Houston, TX 77251
130. Andrew B. White, Los Alamos National Laboratory, P. O. Box 1663, MS-265, Los Alamos, NM 87545
131. David L. Williamson, National Center for Atmospheric Research, P. O. Box 3000, Boulder, CO 80307
132. Samuel Yee, Air Force Geophysics Lab, Department LYP, Hancom AFB, Bedford, MA 01731

133. Office of Assistant Manager for Energy Research and Development, U.S. Department of Energy, Oak Ridge Operations Office, P. O. Box 2001, Oak Ridge, TN 37831-8600

134-143. Office of Scientific & Technical Information, P. O. Box 62, Oak Ridge, TN 37831