INSTITUTE OF PHYSICS PUBLISHING SMART MATERIALS AND STRUCTURES
Smart Mater. Struct. **14** (2005) 895–903 doi:10.1088/0964-1726/14/5/026

Signal identification in smart composite materials using the two-dimensional fast Fourier transform

Kevin Loewke[1], David Meyer[2], Anthony Starr[1] and Sia Nemat-Nasser[1]

[1] Center of Excellence for Advanced Materials, Department of Mechanical and Aerospace Engineering, University of California, San Diego, La Jolla, CA 92093-0416, USA
[2] Department of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112, USA

E-mail: kloewke@ucsd.edu, dmeyer@math.ucsd.edu, astarr@ucsd.edu and sia@ucsd.edu

Received 17 September 2004, in final form 17 March 2005
Published 2 September 2005
Online at stacks.iop.org/SMS/14/895

Abstract
This work is part of an effort to structurally integrate self-sensing functionality into smart composite materials using embedded microsensors and local network communication nodes. Here we address the issue of data management through the development of localized processing algorithms. We demonstrate that the two-dimensional fast Fourier transform (FFT) is a useful algorithm due to its hierarchical structure and ability to determine the relative magnitudes of different spatial wavelengths in a material. This may be applied, for example, to determine the global components of a strain field or temperature distribution. We develop two methods for implementing the distributed 2D FFT based on the radix-2 (row–column) and radix-2 $\times$ 2 (vector–radix) structures, and compare them in terms of computational requirements within a low power, low bandwidth network of microprocessors. Our results show that the vector–radix algorithm requires 50% fewer multiplications than the row–column algorithm when performed in a distributed manner. Since the most important information of the 2D FFT can often be found in the lowest frequency components, we develop pruning methods for the distributed row–column and vector–radix algorithms that reduce internode communication requirements by 50% in both cases. We conclude that the pruned version of the distributed vector–radix 2D FFT is the most efficient of the methods investigated for rapid signal identification in smart composite materials.

1. Introduction

Composite materials have found a wide range of applications in engineering due to their high strength-to-weight ratio, resistance to fatigue, and low thermal expansion. Composites present challenges for damage detection, however, since much of the damage is often interlaminar and not readily detectable [1]. Furthermore, inspection usually takes place after the damage has already occurred, leaving the inspection process to look for residual signs of the failure condition. There is therefore a current need to develop real time and *in situ* health monitoring techniques that enable a rapid assessment of material's state of health.

There have been a number of efforts aimed at incorporating non-structural elements into composite materials for damage detection and assessment [2–4]. The work described here is part of an ongoing effort to develop a new type of smart composite material that can monitor its own health using a high density of embedded microsensors and local network communication nodes [5]. Integrating such devices will allow the material to internally acquire and process structural information. Unlike global networks

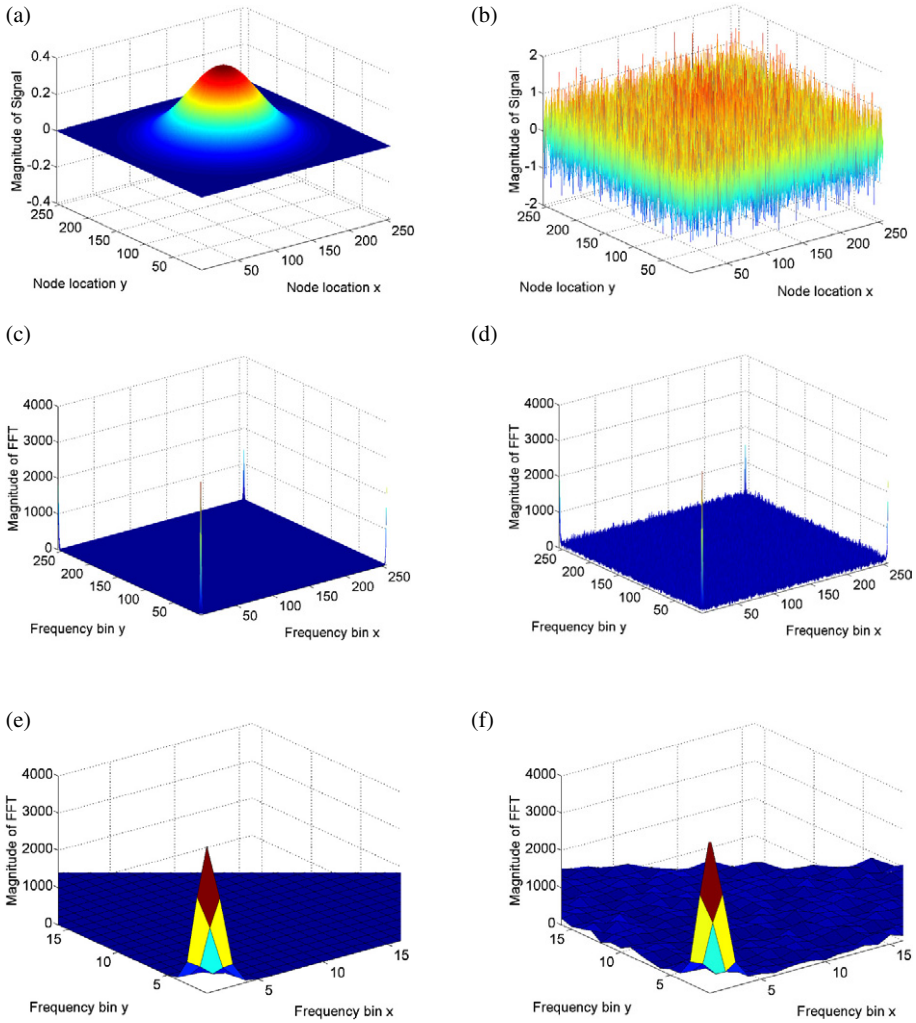

Figure 1. Application of the 2D FFT to a simulated temperature signal of size 256 × 256, with and without added noise. By analyzing only a few output points, we can determine that the signal is dominated in both the x-direction and y-direction by the first frequency components, indicating a temperature peak. These simulations also show that for some situations the 2D FFT is capable of retrieving signals in noisy environments. (a) Simulated signal of size 256 × 256, (b) simulated signal shown in (a) with added Gaussian random noise, (c) magnitude of 2D FFT of signal without noise, (d) magnitude of 2D FFT of signal with noise, (e) low frequency components of 2D FFT of signal without noise, and (f) low frequency components of 2D FFT of signal with noise.

that allow only one transmission at a time, local networks exchange information locally between nodes, and allow an increase in total instantaneous bandwidth at the expense of reduced reach. Some of the challenges involved in this project include mechanical integration, electronics requirements and limitations, and sensor selection. This paper addresses the issue of data management.

As the number of sensor nodes increases, the ability to efficiently manage data becomes an important issue. For large structures, shuffling the data from the embedded network to an external processor would require unreasonable bandwidth. Furthermore, we want to take advantage of the available high instantaneous bandwidth within the local network. It is therefore necessary to develop efficient localized processing algorithms that are hierarchical in structure and can be easily distributed across the network. This paper investigates the implementation of the two-dimensional fast Fourier transform (FFT) as one such algorithm. The 2D FFT essentially decomposes a discrete signal into its frequency components (of varying magnitude), and shuffles the low frequency components to the corners. That is, it reveals the relative magnitudes of different spatial wavelengths in a material. This may be applied, for example, to determine the global components of a strain field or temperature distribution.

Figure 1 demonstrates the application of the 2D FFT to a simulated signal of size 256 × 256 that could represent a local peak in temperature in a composite material. The magnitude of the 2D FFT is plotted with respect to spatial frequency. It is immediately clear that the most significant information exists in the lowest frequency components. The zeroth frequency has the largest magnitude, and represents the constant function in the signal. By analyzing only a few other output points, we can determine that the signal is dominated in both the x-direction and y-direction by the first frequency components. This indicates that the signal consists of a single temperature peak, as opposed to several peaks, or a gradient in one direction.

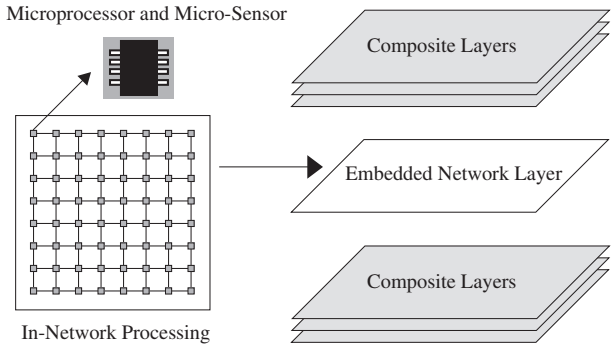

Figure 2. Integration of network nodes in a composite plate.

Figure 1 also shows that the 2D FFT is capable of revealing these characteristics when Gaussian random noise is added to the signal. Such noise could be attributed to small temperature fluctuations that are the result of atmospheric (or other external) conditions. This property is especially important as most practical monitoring would occur in a noisy environment.

The focus of this paper is to establish an efficient algorithm for implementing the 2D FFT using a network of nodes embedded within a composite material. The implementation of the 2D FFT using more than one processor has been widely studied [6–8]. In most cases the number of processors is significantly less than the number of input data points, and the 2D FFT algorithms are said to be performed in a *parallel* manner. In this paper, however, we will consider the implementation of the 2D FFT using a network where the number of nodes is the same as the number of input data points. We will refer to these algorithms as *distributed*. Such an implementation requires that some primitive processing capability be introduced at each node through the inclusion of a microprocessor. We therefore must outline specific assumptions regarding the processing capabilities of the network, which is shown graphically in figure 2:

- The data sequence $x(k_1, k_2)$, defined over $0 \leqslant k_1 < N$, $0 \leqslant k_2 < N$, is acquired and processed by an $N \times N$ array of nodes.
- Each node contains a microsensor and a microprocessor with limited capabilities.
- Each microsensor is wired to its own microprocessor, and each microprocessor is then wired only to its nearest neighbors (up, down, left, right).

While these limitations are not applicable in all scenarios, they apply specifically to the technology involved in this project [5], and are compatible with the low power, low bandwidth networks that can be embedded in composite materials. Although each microprocessor is limited as postulated, the total processing power is significant when the algorithms are performed in a distributed manner. In the following sections, we develop two algorithms for the 2D FFT based on the radix-2 (row–column) and the radix-2 $\times$ 2 (vector–radix) methods [9], where the term *radix* refers to the size of the FFT decomposition. We then develop the distributed versions of both algorithms, and compare their computational requirements.

Table 1. Bit-reversal process for a sequence of size $N = 8$.

Index	Binary	Bit-reversed binary	Bit-reversed index
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

2. The fast Fourier transform

We begin with a brief review of the FFT, which is an efficient algorithm used to implement the discrete Fourier transform (DFT). If we define the complex number

$$W_N = \mathrm{e}^{-2\pi \mathrm{j}/N}, \tag{1}$$

then for a given sequence x_k, defined over $0 \leqslant k < N$, we can define its one-dimensional DFT, X_n, as the following sequence of complex numbers:

$$X_n = \sum_{k=0}^{N-1} x_k W_N^{nk}, \tag{2}$$

where $0 \leqslant n < N$. Using what is sometimes called the *Danielson–Lanczos lemma*, the DFT of size N can be rewritten as the sum of two DFTs of size $N/2$. One is formed from the even-numbered components, and the other from the odd-numbered components [10]:

$$\begin{aligned} X_n &= \sum_{k=0}^{N-1} x_k \mathrm{e}^{-2\pi \mathrm{j} kn/N} \\ &= \sum_{k=0}^{N/2-1} x_{2k} \mathrm{e}^{-2\pi \mathrm{j}(2k)n/N} + \sum_{k=0}^{N/2-1} x_{2k+1} \mathrm{e}^{-2\pi \mathrm{j}(2k+1)n/N} \\ &= \sum_{k=0}^{N/2-1} x_{2k} \mathrm{e}^{-2\pi \mathrm{j} kn/(N/2)} + W_N^k \sum_{k=0}^{N/2-1} x_{2k+1} \mathrm{e}^{-2\pi \mathrm{j} kn/(N/2)} \\ &= X_k^e + W_N^k X_K^o. \end{aligned} \tag{3}$$

The procedure is applied recursively until the data set is reduced to transforms of only two points. This is the basis for the FFT, which can be represented graphically using the radix-2 butterfly shown in figure 3. The recursive decomposition of the sequence into even and odd components requires a bit-reversal reordering to take place at the beginning of the butterfly. Bit-reversal reordering is achieved for a sequence by performing the following steps for each number: transform its index into a binary representation, reverse the order of bits, and transform back to the appropriate index. Table 1 shows this process for a sequence of size $N = 8$. The 1D FFT can therefore be achieved by bit-reversing the input sequence, and then calculating transforms of length $2, 4, 8, \ldots, N$ using a radix-2 butterfly network.

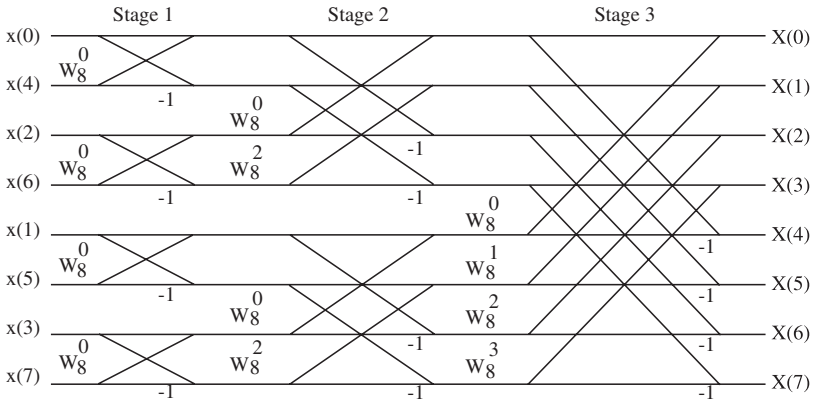

Figure 3. Radix-2 FFT butterfly for size $N = 8$. The constants (powers of W and -1) multiply the number propagating from left to right along the adjacent line. A junction where lines converge indicates an addition. For example, at the end of the first stage, we have $x(0) \leftarrow x(0) + W_8^0 x(4)$.

3. 2D FFT algorithms

3.1. Radix-2 row–column FFT

For a given array $x(k_1, k_2)$, defined over $0 \leqslant k_1 < N_1$, $0 \leqslant k_2 < N_2$, we can define its two-dimensional DFT, $X(n_1, n_2)$, as the following array of complex numbers:

$$X(n_1, n_2) = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} x(k_1, k_2) W_{N_1}^{n_1 k_1} W_{N_2}^{n_2 k_2}, \tag{4}$$

where $0 \leqslant n_1 < N_1$, $0 \leqslant n_2 < N_2$. It is assumed that N_1 and N_2 are the same power of 2, $N_1 = N_2 = 2^M$. The 2D DFT can be efficiently implemented using the row–column FFT, in which 1D FFTs are sequentially performed on each row and then each column of the original sequence $x(k_1, k_2)$. The order of operations is then the following: first, bit-reverse each row, and calculate transforms of length $2, 4, 8, \ldots, N$ (using the radix-2 butterfly network) for each row; second, bit-reverse each column, and calculate transforms of length $2, 4, 8, \ldots, N$ (using the radix-2 butterfly network) for each column.

3.2. Radix-2 × 2 vector–radix FFT

The 2D DFT (4) can also be efficiently implemented using the vector–radix 2D FFT. The vector–radix algorithm involves a decomposition of the 2D DFT into sums of smaller 2D DFTs, until only DFTs of size 2×2 remain. Figure 4 shows the radix-2×2 butterfly, which is based on the following decomposition [9]:

$$\begin{aligned}
X(n_1, n_2) &= G_{00}(n_1, n_2) + W_N^{k_2} G_{01}(n_1, n_2) \\
&\quad + W_N^{k_1} G_{10}(n_1, n_2) + W_N^{k_1+k_2} G_{11}(n_1, n_2) \\
X(n_1 + N/2, n_2) &= G_{00}(n_1, n_2) + W_N^{k_2} G_{01}(n_1, n_2) \\
&\quad - W_N^{k_1} G_{10}(n_1, n_2) - W_N^{k_1+k_2} G_{11}(n_1, n_2) \\
X(n_1, n_2 + N/2) &= G_{00}(n_1, n_2) - W_N^{k_2} G_{01}(n_1, n_2) \\
&\quad + W_N^{k_1} G_{10}(n_1, n_2) - W_N^{k_1+k_2} G_{11}(n_1, n_2) \\
X(n_1 + N/2, n_2 + N/2) &= G_{00}(n_1, n_2) - W_N^{k_2} G_{01}(n_1, n_2) \\
&\quad - W_N^{k_1} G_{10}(n_1, n_2) + W_N^{k_1+k_2} G_{11}(n_1, n_2),
\end{aligned} \tag{5}$$

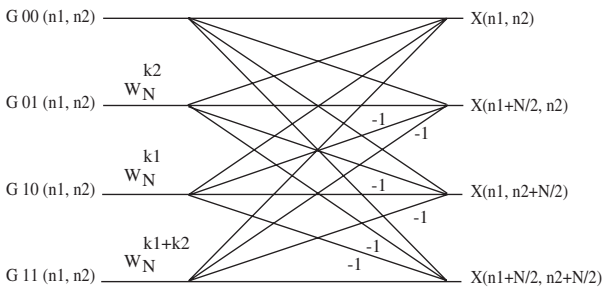

Figure 4. Radix-2 × 2 FFT butterfly.

where the G terms are calculated as

$$\begin{aligned}
G_{00}(n_1, n_2) &= \sum_{k_1=0}^{N/2-1} \sum_{k_2=0}^{N/2-1} x(2k_1, 2k_2) W_N^{2n_1 k_1} W_N^{2n_2 k_2} \\
G_{01}(n_1, n_2) &= \sum_{k_1=0}^{N/2-1} \sum_{k_2=0}^{N/2-1} x(2k_1, 2k_2 + 1) W_N^{2n_1 k_1} W_N^{2n_2 k_2} \\
G_{10}(n_1, n_2) &= \sum_{k_1=0}^{N/2-1} \sum_{k_2=0}^{N/2-1} x(2k_1 + 1, 2k_2) W_N^{2n_1 k_1} W_N^{2n_2 k_2} \\
G_{11}(n_1, n_2) &= \sum_{k_1=0}^{N/2-1} \sum_{k_2=0}^{N/2-1} x(2k_1 + 1, 2k_2 + 1) W_N^{2n_1 k_1} W_N^{2n_2 k_2}.
\end{aligned} \tag{6}$$

The order of operations for the vector–radix algorithm is summarized by the following: first, bit-reverse all rows and then all columns; second, calculate transforms of size $2 \times 2, 4 \times 4, 8 \times 8, \ldots, N \times N$ using the radix-2×2 butterfly.

The key part of the vector–radix algorithm is that repetitive multiplications can be eliminated by using the identity $W_N^{k_1} W_N^{k_2} = W_N^{k_1+k_2}$. For example, at the last (third) stage of the vector–radix algorithm for a sequence of size 8×8, instead of multiplying each row and each column by

$$(1 \quad 1 \quad 1 \quad 1 \quad W_8^0 \quad W_8^1 \quad W_8^2 \quad W_8^3),$$

the entire matrix is multiplied by

$$\begin{pmatrix} 1 & 1 & 1 & 1 & W_8^0 & W_8^1 & W_8^2 & W_8^3 \\ 1 & 1 & 1 & 1 & W_8^0 & W_8^1 & W_8^2 & W_8^3 \\ 1 & 1 & 1 & 1 & W_8^0 & W_8^1 & W_8^2 & W_8^3 \\ 1 & 1 & 1 & 1 & W_8^0 & W_8^1 & W_8^2 & W_8^3 \\ W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^1 & W_8^2 & W_8^3 \\ W_8^1 & W_8^1 & W_8^1 & W_8^1 & W_8^1 & W_8^2 & W_8^3 & W_8^4 \\ W_8^2 & W_8^2 & W_8^2 & W_8^2 & W_8^2 & W_8^3 & W_8^4 & W_8^5 \\ W_8^3 & W_8^3 & W_8^3 & W_8^3 & W_8^3 & W_8^4 & W_8^5 & W_8^6 \end{pmatrix}.$$

When implemented on a single processor, the vector–radix algorithm requires 25% fewer multiplications than the row–column algorithm [11].

3.3. Relation between the row–column and vector–radix algorithms

The vector–radix algorithm can be understood as a variation of the row–column FFT with two main differences: first, the vector–radix algorithm applies the bit-reversal reordering to all rows *and* all columns before any butterfly operations are performed; second, at each stage of the radix-2 butterfly (shown in figure 3), the vector–radix algorithm performs both row-wise *and* column-wise transforms before proceeding to the next stage. The order of operations for the vector–radix FFT can therefore be restated as the following: first, bit-reverse all rows and then all columns; second, calculate, in turn, row-wise and column-wise transforms of length 2, 4, 8, ..., N using the radix-2 butterfly network. As before, the vector–radix algorithm eliminates repetitive multiplications at each stage by performing a matrix multiplication before the row and column additions. We will use this version to develop the distributed implementation of the vector–radix algorithm.

4. Distributed 2D FFT

The row–column and vector–radix FFT algorithms have been derived assuming that all calculations would occur on a single processor. The purpose of this paper, however, is to develop distributed algorithms suitable for implementation in a network of microprocessors. Since the 2D FFT has a hierarchical structure, and since butterfly operations can be performed independently and concurrently on separate processors, it is a natural distributed algorithm candidate. For example, the row–column algorithm performs butterfly operations on each row and then each column of the sequence. This process can be distributed across an array of microprocessors since each row and each column can independently and concurrently perform their own butterfly operations. When performed concurrently, this requires no extra cost over the requirements of a single row performing the operations.

Using the assumptions outlined in section 1, we can determine the computation required to perform the distributed algorithms using an array of microprocessors. We evaluate these requirements in terms of communication and calculation *steps*. One calculation step includes all concurrent additions or multiplications within the entire microprocessor array. In the same manner, one communication step includes all possible concurrent data transfers (from one node to its appropriate neighbor) within the entire microprocessor array.

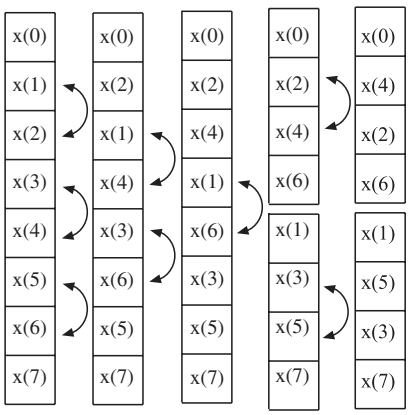

Figure 5. Bit-reversal swapping method for $N = 8$.

We outline the basic communication steps assuming that a node may either send data to, or receive data from, one neighbor at a time. While we develop and compare the numerical value of communication steps, the actual required communication time would be determined by the internal processor clock rate. Furthermore, when physically constructing the communication networks, it is likely that the bandwidth of the network would be increased through the inclusion of more wires between nodes, or the ability of a node to communicate with more than one neighbor at a time. In these cases, the communication requirements would be reduced by a constant factor corresponding to the increase in bandwidth.

4.1. Communication requirements

Communication between nodes takes place during two procedures: bit-reversals and butterfly networks. A one-dimensional bit-reversal can be performed using a convenient swapping method, where adjacent nodes swap their respective values. The method follows a triangular pattern for the inner $N - 2$ nodes, divides the sequence in half, and continues recursively for a total of $\log_2 N - 1$ times. Figure 5 demonstrates this method for a sequence of size $N = 8$. Since all swaps within an array can be performed concurrently at a cost of two data transfers, the number of communication steps required to complete the bit-reversal in one dimension is

$$\begin{aligned} \text{Steps} &= 2\sum_{i=1}^{\log_2 N}\left(\frac{N}{2^i} - 1\right) \\ &= 2N\sum_{i=1}^{\log_2 N}\frac{1}{2^i} - 2\sum_{i=1}^{\log_2 N} 1 \\ &\approx 2N - 2\log_2 N, \end{aligned} \tag{7}$$

for large N. Section 3.3 shows that the row–column and vector–radix algorithms both implement the radix-2 butterfly, except in different orders. The number of communication steps required by the two algorithms is therefore identical, and we only need to establish the method of data transfer for the radix-2 butterfly, which is demonstrated in figure 6 for size $N = 8$. The data transfer is straightforward and strictly follows the assumptions in section 1. During each stage of data transfer, all microprocessors retain their original value to use in the

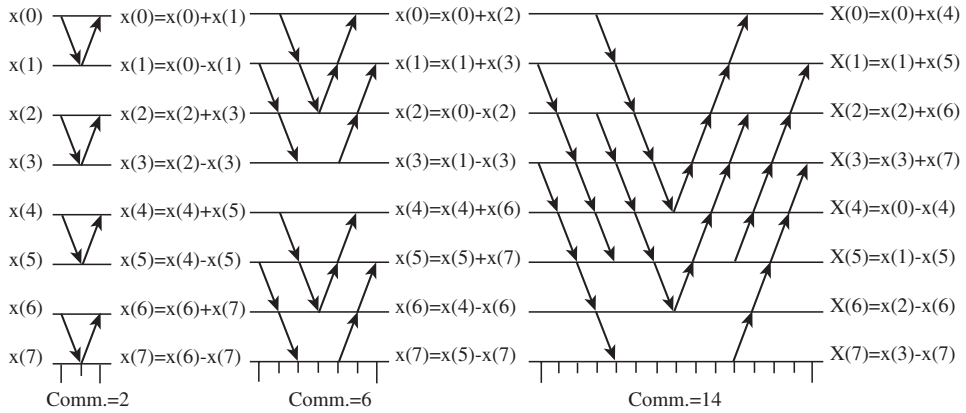

Figure 6. Data transfer among an array of $N = 8$ microprocessors required for radix-2 butterfly network, assuming already bit-reversed input. (Multiplications by W are omitted for simplicity.) The number of communications for each stage is indicated at the bottom. Distributing the 2D FFT allows for all rows or all columns to simultaneously perform the butterfly operations.

butterfly calculation. The number of communication steps required for a radix-2 butterfly of size N is

$$\begin{aligned}\text{Steps} &= 2\sum_{i=1}^{\log_2 N}\left(\frac{2N}{2^i}-1\right)\\ &= 4N\sum_{i=1}^{\log_2 N}\frac{1}{2^i}-2\sum_{i=1}^{\log_2 N}1\\ &\approx 4N-2\log_2 N, \qquad (8)\end{aligned}$$

for large N. We can now determine the total number of communication steps required for both the distributed row–column and vector–radix algorithms. Counting the bit-reversal twice and the radix-2 butterfly twice (once for all columns and once for all rows) yields, for large N,

$$\begin{aligned}\text{Steps} &\approx 2(2N-2\log_2 N)+2(4N-2\log_2 N)\\ &= 12N-8\log_2 N. \qquad (9)\end{aligned}$$

4.2. Calculation requirements

Calculations for the distributed row–column and vector–radix algorithms can also be done in a distributed manner, such that at each stage of the radix-2 butterfly all microprocessors concurrently perform a single complex addition or multiplication. (Some multiplications are by unity.) In our analysis, one calculation step will include all possible concurrent multiplications or additions within the microprocessor array. The row–column algorithm performs multiplications at the beginning, and additions at the end, of each stage of the radix-2 butterfly. Applying the radix-2 butterfly twice (once for all rows and once for all columns) requires $2\log_2 N$ addition steps and $2\log_2 N$ multiplication steps.

At each stage of the radix-2 butterfly, the vector–radix algorithm performs both row-wise and column-wise transforms before proceeding to the next stage. Although performed in a different order, the vector–radix algorithm requires the same number of addition steps as the row–column algorithm. Since the vector–radix algorithm performs a single matrix multiplication at the beginning of each stage, however, it requires only $\log_2 N$ multiplication steps. The vector–radix algorithm therefore requires 50% fewer multiplications than the row–column algorithm when performed in a distributed manner.

5. Pruning the 2D FFT

As shown in figure 1, the 2D FFT reveals the relative strengths of periodic signals within a composite material. When looking for signals such as large pressure gradients or peaks, the most important information can often be found in the lowest frequency components of the 2D DFT. A significant amount of the output data can therefore be ignored, allowing for modification of the 2D FFT algorithms to improve computational efficiency.

Figure 7 compares the pruned 2D FFT outputs of three simulated temperature signals of size 256×256. By analyzing just a few of the low frequency components of the 2D FFTs, we can easily distinguish between the three signals. Let us consider the gradient signal shown in figure 7(a). The 2D FFT of the gradient shows that the frequency components in the y-direction have a magnitude of close to zero, indicating that the spatial wavelengths in that direction are relatively constant. The x-direction, however, is clearly dominated by the first frequency component. We can conclude from this information that the signal consists of a gradient in the x-direction. Next let us consider the single-peak and double-peak signals shown in figures 7(c) and (e), respectively. The 2D FFTs of the two signals are similar except for one significant difference: the 2D FFT of the double-peak signal is dominated in the x-direction by the second frequency component rather than the first frequency component. This indicates the presence of two peaks in the x-direction, as opposed to one peak.

Pruning techniques [9, 12] can be applied to both the row–column and vector–radix algorithms to eliminate computation that is not necessary for the desired output points. Once the output points are selected, it is convenient to trace the radix-2 butterfly backwards in order to determine which input points are necessary for each stage. As an example, we will consider a pruning algorithm that calculates the 4×4 output matrix $X(n_1, n_2)$, where $0 \leqslant n_1 < 3$, $0 \leqslant n_2 < 3$. Figure 8 shows the necessary communication and calculation steps required for the $N = 8$ radix-2 butterfly. The solid lines are required to complete the butterfly, while dashed lines are ignored. For sequences larger than size $N = 8$, the same results can be achieved by just calculating the first four components at each incremental stage of the radix-2 butterfly. The desired 4×4

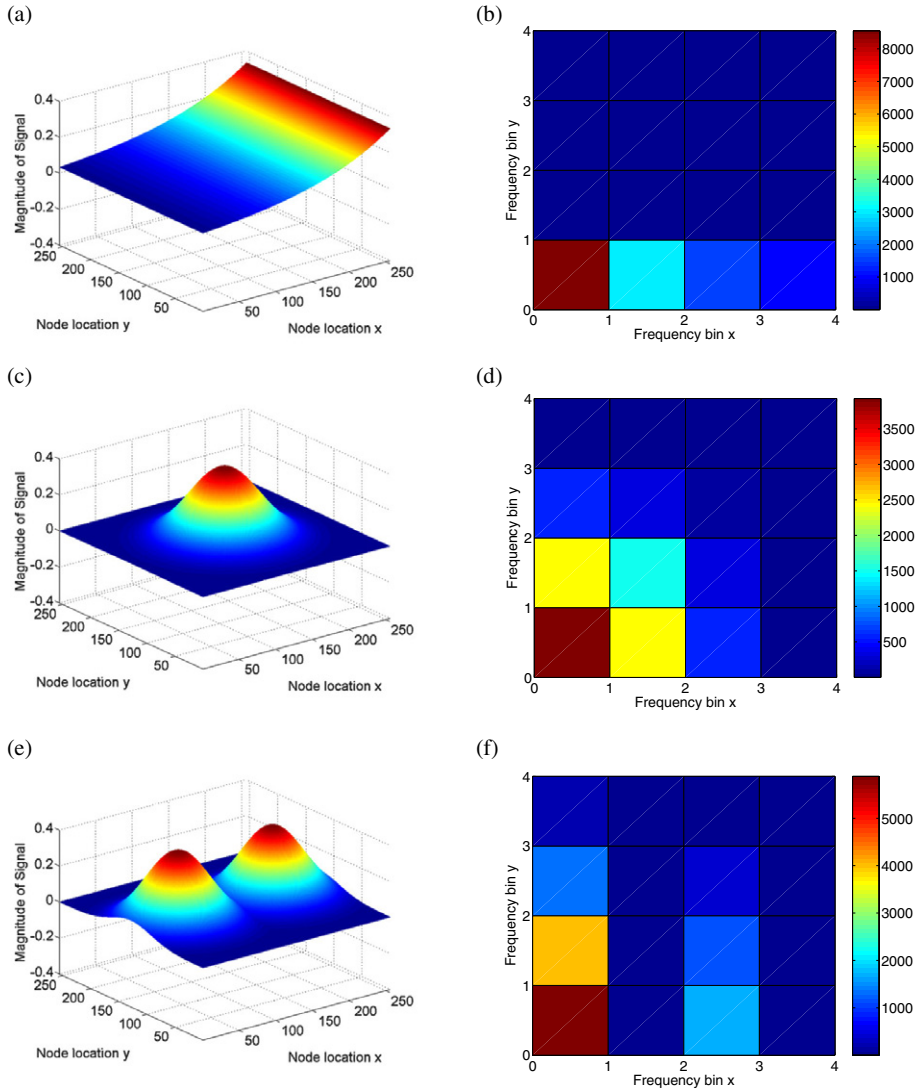

Figure 7. Application of the 2D FFT to three simulated temperature signals of size 256 × 256. Only the 4 × 4 matrix of low frequency components is needed to distinguish between signals that consist of a gradient, one peak, or two peaks. (a) Simulated gradient signal of size 256 × 256, (b) magnitude of 2D FFT of signal in (a), pruned to size 4 × 4, (c) simulated single-peak signal of size 256 × 256, (d) magnitude of 2D FFT of signal in (c), pruned to size 4 × 4, (e) simulated double-peak signal of size 256 × 256, and (f) magnitude of 2D FFT of signal in (e), pruned to size 4 × 4.

output matrix is then achieved by performing the pruned radix-2 butterfly on all rows and columns (using either the row–column or vector–radix algorithms).

5.1. Computational savings from pruning

We established in section 4.1 that the distributed row–column and vector–radix algorithms require the same number of communication steps. This remains true for their pruned versions, as long as the two algorithms produce the same size output. We therefore only need to examine pruning of the radix-2 butterfly network which is used (in different orders) by both algorithms.

When performed in a distributed manner, this method of pruning the 2D FFT does not reduce the number of calculation steps required. Several microprocessors are simply inactive while others within the matrix concurrently perform their own addition or multiplication. A significant amount of communication, however, can be eliminated by pruning. The pruned radix-2 butterfly of figure 8 shows that after the first two stages of transforms, the communication is reduced to sending groups of only four complex numbers at each incremental stage. The number of communication steps required for a pruned radix-2 butterfly of size N (neglecting the first two stages) is

$$\begin{aligned}\text{Steps} &= \sum_{i=1}^{\log_2 N-2}\left(\frac{N}{2^i}+3\right)\\ &= N\sum_{i=1}^{\log_2 N-2}\frac{1}{2^i}+3\sum_{i=1}^{\log_2 N-2}1\\ &\approx N+3\log_2 N, \end{aligned} \tag{10}$$

for large N. The total number of communication steps can now be established for the pruned row–column and vector–radix algorithms. Counting the bit-reversal twice and the pruned

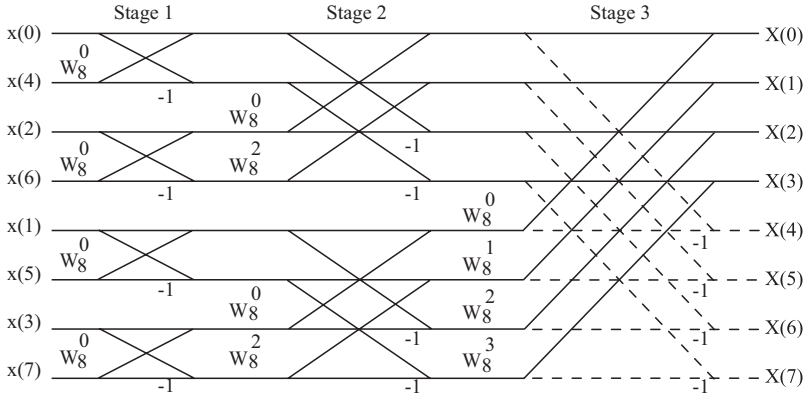

Figure 8. Pruned radix-2 FFT butterfly network for $N = 8$. Solid lines are required, while dashed lines are ignored.

radix-2 butterfly twice (once for all columns and once for all rows) yields, for large N,

$$\begin{aligned}\text{Steps} &\approx 2(2N - 2\log_2 N) + 2(N + 3\log_2 N)\\ &= 6N + 2\log_2 N. \end{aligned} \tag{11}$$

Comparing the number of pruned communication steps (11) to the number of conventional communication steps (9), and taking dominating terms, yields

$$\begin{aligned}\frac{\text{Pruned}}{\text{Conventional}} &\approx \frac{6N + 2\log_2 N}{12N - 8\log_2 N}\\ &\approx \mathrm{O}\left(\tfrac{1}{2}\right). \end{aligned} \tag{12}$$

When compared to the conventional row–column and vector–radix algorithms, the pruned version reduces communication requirements by 50%.

6. Conclusions

This paper investigates the implementation of localized processing algorithms for signal identification in smart composite materials. Our simulations show that the 2D FFT can be a useful tool in revealing the relative magnitudes of different spatial wavelengths of a signal in a material. We develop two distributed methods for implementing the 2D FFT based on the radix-2 (row–column) and radix-2 $\times$ 2 (vector–radix) structures. Results show that the vector–radix 2D FFT requires 50% fewer multiplications than the row–column 2D FFT when performed in a distributed manner. Since only a few output points of the 2D FFT are often desired, we develop pruning techniques that reduce communication by 50% in both cases. We conclude that the pruned version of the distributed vector–radix 2D FFT is the most efficient of the methods investigated for rapid signal identification in composite materials, when the nature of the signal can be inferred from low frequency components.

While the purpose of this paper has been to investigate distributed algorithm techniques for smart composite materials, ultimately we plan to apply this research towards the structural health monitoring of composite materials. The application of the techniques presented in this paper would depend, however, on the size of the embedded network, the spatial density of the network nodes, and the sensor types used. Furthermore, structural health could only be established by including a structural model against which to compare. If these items were adequately addressed, the distributed algorithm techniques presented in this paper could have significant potential for rapidly assessing the health of composite materials.

Future work in this project includes the development of other applications of the 2D FFT such as identifying the location of failures in the material using the correlation of an anticipated failure mode pattern with the 2D FFT of the sampled data set. Future work also includes developing distributed versions of other algorithms useful for signal identification and/or structural health monitoring in composite materials. Ultimately we plan to implement these algorithms experimentally. We have undertaken the initial steps needed to verify the feasibility of embedding sensors in composites, and are currently preparing a composite panel with an embedded array of microsensors and microprocessors.

Acknowledgments

This research was conducted at the Center of Excellence for Advanced Materials (CEAM), University of California, San Diego. Important contributions by D Smith, Y Huang, P Rye, K Schaaf, J Aller, T Plaisted, B Cook, J Isaacs and D Lischer are acknowledged. This work was supported in part by NSF-CMS grant number 0330450.

References

[1] Kessler S S, Spearing S M and Soutis C 2002 Damage detection in composite materials using Lamb wave methods *Smart Mater. Struct.* **11** 269–78

[2] Varadan V K and Vardin V V 2000 Microsensors, microelectromechanical systems (MEMS), and electronics for smart structures and systems *Smart Mater. Struct.* **9** 953–72

[3] Zhou G and Sim L M 2002 Damage detection and assessment in fibre-reinforced composite structures with embedded fibre optic sensors—review *Smart Mater. Struct.* **11** 925–39

[4] Watkins S E, Sanders G W, Akhavan F and Chandrashekhara K 2002 Modal analysis using fiber optic sensors and neural networks for prediction of composite beam delamination *Smart Mater. Struct.* **11** 489–95

[5] Starr A F, Nemat-Nasser S, Smith D R and Meyer D A 2003 Development of embedded microelectronic sensor networks in composite materials *Proc. Int. Workshop on Advanced Sensors, Structural Health Monitoring and Smart Structures (Yokohama, Japan)* pp 1–6

[6] Nolle M and Jungclaus N 1994 Efficient implementation of FFT-like algorithms on MIMD systems *Proc. EUSIPCO-94, 7. Europ. Sign. Proc. Conf.* vol 3, pp 1625–8

[7] Balducci M, Choudary A and Hamaker J 1996 Comparative analysis of FFT algorithms in sequential and parallel form *Mississippi State University Conf. on Digital Signal Processing* pp 5–16

[8] Baptist L M and Cormen T H 1999 Multidimensional, multiprocessor, out-of-core FFTs with distributed memory and parallel disks *ACM Symp. on Parallel Algorithms and Architectures Archive, Proc. 11th Annual ACM Symp. on Parallel Algorithms and Architectures* pp 242–50

[9] Knudsen K S and Bruton L T 1993 Recursive pruning of the 2-D DFT with 3-D signal processing applications *IEEE Trans. Signal Process.* **41** 1340–56

[10] Press W H, Flannery B P, Teukolsky S A and Vetterling W T 1992 *Numerical Recipes in C: The Art of Scientific Computing* 2nd edn (Cambridge: Cambridge University Press) pp 504–23

[11] Chu C 1988 Comparison of two-dimensional FFT methods on the hypercube *The 3rd Conf. on Hypercube Concurrent Computers and Applications* vol 2, pp 1430–7

[12] Markel J 1971 FFT pruning *IEEE Trans. Audio Electroacoust.* **19** 305–11