GEOPHYSICS, VOL. 63, NO. 1 (JANUARY-FEBRUARY 1998); P. 279–288, 9 FIGS., 2 TABLES.

Parallel 3-D pseudospectral simulation of seismic wave propagation

Takashi Furumura*, B. L. N. Kennett‡, and Hiroshi Takenaka**

ABSTRACT

Three-dimensional pseudospectral modeling for a realistic scale problem is still computationally very intensive, even when using current powerful computers. To overcome this, we have developed a parallel pseudospectral code for calculating the 3-D wavefield by concurrent use of a number of processors. The parallel algorithm is based on a partition of the computational domain, where the field quantities are distributed over a number of processors and the calculation is concurrently done in each subdomain with interprocessor communications. Experimental performance tests using three different styles of parallel computers achieved a fairly good speed up compared with conventional computation on a single processor: maximum speed-up rate of 26 using 32 processors of a Thinking Machine CM-5 parallel computer, 1.6 using a Digital Equipment DEC-Alpha two-CPU workstation, and 4.6 using a cluster of eight Sun Microsystems SPARC-Station 10 (SPARC-10) workstations connected by an Ethernet. The result of this test agrees well with the performance theoretically predicted for each system. To demonstrate the feasibility of our parallel algorithm, we show three examples: 3-D acoustic and elastic modeling of fault-zone trapped waves and the calculation of elastic wave propagation in a 3-D syncline model.

INTRODUCTION

The pseudospectral method (e.g., Kosloff and Baysal, 1982) is an attractive alternative to other numerical modeling schemes, such as the finite-difference or the finite-element methods, which have been used typically for modeling of seismic wave propagation in a heterogeneous medium. In the pseudospectral method, the field quantities are expanded in terms of Fourier interpolation polynomials, and the spatial differentiation of the quantities is analytically performed in the wavenumber domain. This accurate spatial differentiation can reduce computer memory and computation time by several orders of magnitude as compared with other numerical methods, such as the finite-difference or the finite-element methods (e.g., Fornberg, 1987; Daudt et al., 1989). Thus the pseudospectral method has been applied to high-resolution forward modeling of exploration experiments using a 2-D elastic model (e.g., Kang and McMechan, 1990) or a 2.5-D elastic model (Furumura and Takenaka, 1996).

However, even when using the pseudospectral method, 3-D elastic modeling has been very expensive, because it requires huge amounts of computer memory and computation time; therefore its application to realistic scale 3-D modeling has been restricted to acoustic wave calculations (e.g., Chen and McMechan, 1993; Reshef et al., 1988a; Huang et al., 1995). To overcome this problem, Reshef et al. (1988b) first attempted a parallel pseudospectral computation to calculate the 3-D elastic wavefield in a reasonable geological scale model by use of a CRAY X-MP supercomputer that has four processors and a solid-state mass storage device. Recently, the efficiency of parallel pseudospectral computing has typically been demonstrated for fast modeling of 1-D waves (Renaut and Woo, 1992), 2-D scalar waves (Sato et al., 1994, 1995) and 2-D visco-acoustic waves (Liao and McMechan, 1993).

The purpose of this paper is to propose an alternative effective parallel pseudospectral algorithm for 3-D wave calculation, which can be implemented on the various parallel platforms, such as distributed-memory or shared-memory parallel computers, multiprocessor computers, or a cluster of workstations connected by a computer network. The parallel code is based on the partition of the computational domain, and the field quantities are assigned to each of a number of processors.

To clarify the parallel pseudospectral algorithm, we first give a brief explanation of a pseudospectral computing scheme for elastic waves with some numerical techniques, such as the incorporation of boundary conditions and the spatial differentiation using the fast Fourier transform (FFT). We then review an alternative fast differentiation scheme using the FFT for

Manuscript received by the Editor September 5, 1995; revised manuscript received February 24, 1997.
*Hokkaido University of Education, Midorigaoka 2-34-1, Iwamizawa, 068, Japan. E-mail: furumura@iwa.hokkyodai.ac.jp.
‡Research School of Earth Sciences, Australian National University, Canberra ACT 0200, Australia. E-mail: brian@rses.anu.edu.au.
**Department of Earth and Planetary Sciences, Kyushu University, Hakozaki 6-10-1, Fukuoka, 812-81, Japan. E-mail: takenaka@geo.kyushu-u.ac.jp.

real-valued data (real FFT), which is about twice as fast as the conventional differentiation by the FFT for complex-valued data (complex FFT). Next we present a parallel pseudospectral scheme based on a partition of the computational domain, which is the main part of this paper. The speed-up rate of the parallel performance predicted from the theoretical experiments is confirmed by the benchmark tests using three parallel platforms: Thinking Machines CM-5 parallel computer, Digital Equipment DEC-Alpha multi-CPU workstation, and a cluster of Sun Microsystems SPARC-10 workstations connected by an Ethernet network. Finally, we show some examples calculated by the present parallel pseudospectral scheme to demonstrate the feasibility of our parallel algorithm implemented on current parallel computational environments for realistic 3-D elastic modeling.

3-D PSEUDOSPECTRAL MODELING FOR ELASTIC WAVES

In this section, we review briefly a pseudospectral modeling approach for calculating 3-D elastic waves. In a 3-D rectangular system, with x and y as the horizontal coordinates and z as the vertical coordinate, the equation of motion is represented as

$$\rho \ddot{U}_p = \frac{\partial \sigma_{xp}}{\partial x} + \frac{\partial \sigma_{yp}}{\partial y} + \frac{\partial \sigma_{zp}}{\partial z} + f_p, \qquad (p = x, y, z), \quad (1)$$

where σ_{pq} $(p, q = x, y, z)$ are stress components, f_p are body forces, $\ddot{U}_p$ are the second partial-time derivatives of the displacement components (i.e., acceleration components), and ρ is the density. In an isotropic elastic medium, the stress components are given by

$$\sigma_{pq} = \lambda\,(e_{xx}+e_{yy}+e_{zz})\delta_{pq} + 2\mu\, e_{pq}, \qquad (p, q = x, y, z), \quad (2)$$

where λ and μ are the Lamé constants, e_{pq} are the strain components, and δ_{pq} denotes Kronecker's delta. The strain components are defined as

$$e_{pq} = \frac{1}{2}\left(\frac{\partial U_p}{\partial q} + \frac{\partial U_q}{\partial p}\right), \qquad (p, q = x, y, z), \quad (3)$$

where U_p are the displacement components. In the pseudospectral method, the spatial derivatives in equations (1) and (3) are calculated analytically in the wavenumber domain. For the time evaluation, an explicit scheme is used—the wavefield at the next time step is calculated using the current and previous wavefields. For example, the following second-order finite-difference time-integration scheme is often used:

$$\dot{U}_p^{n+1/2} = \dot{U}_p^{n-1/2} + \ddot{U}_p^n \Delta t, \quad (4)$$

and

$$U_p^{n+1} = U_p^{n-1} + \dot{U}_p^{n+1/2} \Delta t, \quad (5)$$

where $\dot{U}_p$ and U_p $(p = x, y, z)$ denote the particle velocity and displacement, respectively, and Δt is the time step.

Boundary conditions

Since Kosloff and Baysal (1982) first applied the pseudospectral method to seismic wave modeling, a number of numerical techniques have been developed to incorporate the boundary condition and the radiation condition. For example, the free-surface condition is simply incorporated into the calculation by adding a number of zeros to the stress components above the free surface or, equivalently, because of periodicity, below the bottom of the model, prior to vertical differentiations with respect to z. When a seismic point source is placed near the free surface, an oscillating noise known as the Gibbs phenomena often appears in the z-derivatives of the discontinuous data. This noise can be suppressed fairly well by using an alternative differentiation scheme developed by Furumura and Takenaka (1992) ("symmetric differentiation"), in which the data are folded at the free surface prior to the z-differentiation to extinguish the discontinuity as

$$\hat{f}(n\Delta x) = \begin{cases} f(n\Delta x), & 0 \le n \le N-1 \\ f[(2N-n-1)\Delta x], & N \le n \le 2N-1 \end{cases} \quad (6)$$

and then the differentiation for the double-length $(2N)$ data is calculated. The extrapolated parts (i.e., $n = N, \ldots, 2N-1$) are removed immediately after the differentiation and not used in the subsequent calculations.

In the pseudospectral method, artificial reflections do not occur, but wraparound appears from the outer boundaries of the numerical mesh because of the spatial periodicity implicitly involved in the FFT used for calculating the spatial derivatives. To suppress this, the absorbing boundary condition in Cerjan et al. (1985) or Kosloff and Kosloff (1986), based on gradual reduction of the amplitude in a strip of nodes along the boundaries of the mesh, is often applied. Recently, an alternative technique for the wraparound elimination based on a simple modification of the wavefield to cancel the spatial periodicity was developed by Furumura and Takenaka (1995).

Fourier differentiation

A main advantage of the pseudospectral method over the traditional finite-difference method is that the differentiation of the field variables requires fewer grid points per wavelength for a given accuracy (Fornberg, 1987; Daudt et al., 1989). The exact differentiation in the wavenumber domain is calculated efficiently by means of an FFT as follows. First, a field quantity at the spatially discretized locations $f(n\Delta x)$, $(n = 0, 1, \ldots, N-1)$ is transformed to the wavenumber domain by using a 1-D FFT:

$$F(\ell\Delta k) = \Delta x \sum_{n=0}^{N-1} f(n\Delta x) e^{-i2\pi n\ell/N}, \quad (7)$$

where $F(\ell\Delta k)$ $(\ell = 0, 1, \ldots, N-1;\ \Delta k = 2\pi/(N\Delta x))$ represents the Fourier transform of $f(n\Delta x)$. The result is then multiplied by the discrete spatial wavenumbers $\ell\Delta k$ and the imaginary unit i to obtain the derivative in the wavenumber domain and then transformed back to the physical domain using an inverse 1-D FFT, that is,

$$\frac{d}{dx} f(n\Delta x) = \frac{1}{N\Delta x} \sum_{\ell=0}^{N-1} i\,(\ell\Delta k)\, F(\ell\Delta k)\, e^{i2\pi n\ell/N}. \quad (8)$$

For the differentiation of 3-D variables, the calculations of equations (7) and (8) are carried out sequentially along the x-, y-, and z-directions. Thus an enormous number of FFT runs are required for calculating the derivatives appearing in equations (1) and (3) at each time step. For example, a 3-D model with 128^3 grid points requiring the evaluation of the wavefield for

1000 time steps needs 589 824 000 FFT runs in total. Therefore, it is very important to use a fast FFT program when we perform a 3-D pseudospectral modeling.

Differentiation by the real FFT

In the pseudospectral modeling, the FFT for the transform of complex-valued data (i.e., complex FFT) is often used where real-valued data to be differentiated are loaded into the real components of a complex array, while the imaginary components are filled up with zero values. The transform for the real-valued data can also be calculated using both real and imaginary components of one complex array, so two sets of data are efficiently transformed simultaneously (e.g., Zhou, 1992). Such a transformation for real-valued data is also efficiently calculated by simply using the transform for real-valued data such as real FFT (e.g., Bergland, 1968; Sorensen et al., 1987) and the Hartley transform (see e.g., Saatcilar and Ergintav, 1991).

Now we describe an efficient differentiation scheme for real-valued data by using the real FFT. First, the sequence of data $f(\ell\Delta x)$ to be differentiated is expanded in terms of the discrete sine and cosine polynomials using the real FFT as

$$f(n\Delta k) = \sum_{\ell=0}^{N/2} A(\ell\Delta k)\cos(2\pi n\ell/N) + \sum_{\ell=1}^{N/2} B(\ell\Delta k)\sin(2\pi n\ell/N), \qquad (9)$$

where A and B are cosine and sine coefficients as

$$A(\ell\Delta k) = \begin{cases} \dfrac{1}{N}\displaystyle\sum_{n=0}^{N-1} f(n\Delta x)\cos(2\pi n\ell/N), \\ \qquad\qquad \ell = 0, N/2 \\ \dfrac{2}{N}\displaystyle\sum_{n=0}^{N-1} f(n\Delta x)\cos(2\pi n\ell/N), \\ \qquad\qquad \ell = 1, 2, \cdots, N/2-1 \end{cases} \qquad (10)$$

$$B(\ell\Delta k) = \frac{2}{N}\sum_{n=1}^{N-1} f(n\Delta x)\sin(2\pi n\ell/N).$$

Next, the derivative in the transform domain is calculated by multiplying the discrete spatial wavenumbers $\ell\Delta k$ and involving a phase advance of 90° in the equation (or equivalently, exchanging the coefficients A and B by $-B$ and A, respectively), which is then transformed back to the physical domain by means of the inverse real FFT as

$$\frac{d}{dx}f(n\Delta x) = \sum_{\ell=1}^{N/2-1} (\ell\Delta k)B(\ell\Delta k)\cos(2\pi n\ell/N) + \sum_{\ell=1}^{N/2}(-\ell\Delta k)A(\ell\Delta k)\sin(2\pi n\ell/N). \qquad (11)$$

We confirmed that the differentiation using the real FFT pair is twice as fast as the conventional calculation using the complex FFT (Furumura et al., 1993).

PARALLEL PSEUDOSPECTRAL COMPUTING BY DOMAIN PARTITION

In this section, we propose a parallel algorithm for the pseudospectral computation based on a partition of the computational domain. In the parallel pseudospectral computation, each operation is simultaneously done on all subdomains with some interprocessor communications to exchange data between subdomains.

Let NX, NY, and NZ denote the number of grid points in the x-, y-, and z- directions of a 3-D array $DATA$ for a field quantity, which is partitioned into NP equisized subdomain. Figure 1a illustrates an example of the domain partition for four processors ($NP = 4$) model where $DATA$ is partitioned horizontally into four pieces $PDATA^p$ ($p = 1, 2, \ldots, NP$) of size $NX \times NY \times (NZ/NP)$ as follows:

$$PDATA^p\left(1:NX; 1:NY; 1:\frac{NZ}{NP}\right) = DATA\left[1:NX; 1:NY; \frac{NZ}{NP}(p-1)+1:\frac{NZ}{NP}p\right].$$

Each processor calculates the wavefield over each subdomain, concurrently, from equations (1) to (5) excluding the z-differentiation.

For the calculation of the z-derivatives, the field quantity is partitioned vertically into equisized subdomain $ZWORK^p (p = 1, 2, \ldots, NP)$, and the subarray is also mapped onto each processor as

$$ZWORK^p\left(1:\frac{NX}{NP}; 1:NY; 1:NZ\right) = DATA\left[\frac{NX}{NP}(p-1)+1:\frac{NX}{NP}p; 1:NY; 1:NZ\right].$$

Since each processor has only part of $ZWORK^p$ in its own memory, $PDATA^p$ (e.g., the shaded part in Figure 1b), and most of the parts are on the different subdomains, interprocessor communications are required to fill out the $ZWORK^p$. For this purpose each processor sends the portion of data (size $(NX/NP) \times NY \times (NZ/NP)$) to the other $NP-1$ processors and at the same time receives the same size of data from $NP-1$ processors. After all data are collected in the $ZWORK^p$, the z differentiation can be performed. The results are then redistributed over the processors by reversing the sequence of the operations. Because such a data swap with other subdomains only appears in the z-differentiation, the cost of interprocessor communication is expected to be small relative to the total computation time. The algorithm of the z-differentiation with interprocessor communication can be summarized as

```
1) for I := 1 to NP  do  /* Send data to NP-1
                            processors */
     BUFF := PDATA[NX/NP*(I-1)+1:NX/NP*I;
             1:NY; 1:NZ]
     if ( I = p ) then
       ZWORK[1:NX/NP; 1:NY; NZ/NP*(I-1)+1:NZ/NP*I]
       := BUFF
     else
       send BUFF to processor I
     end if
   end do
```

```
2) NR := 1  /* Receive data from NP-1
                        processors */
   while ( NR < NP ) do
     (a) receive data from processor I and store
         into BUFF
     (b) ZWORK[1:NX/NP; 1:NY; NZ/NP*(I-1)+1:NZ/NP*I]
         := BUFF
     (c) NR := NR + 1
   end do
3) perform z differentiation on ZWORK
4) for I := 1 to NP do  /* Send results to NP-1
                           processors */
     BUFF := ZWORK[1:NX/NP; 1:NY;
             NZ/NP*(I-1)+1:NZ/NP*I]
     if ( I = p ) then
       PDATA[NX/NP*(I-1)+1:NX/NP*I; 1:NY; 1:NZ]
       := BUFF
     else
       send BUFF to processor I
   end do
5) NR := 1  /* Receive results from NP-1
               processors */
   while ( NR < NP ) do
     (a) receive data from processor I and store
         into BUFF
     (b) PDATA[NX/NP*(I-1)+1:NX/NP*I; 1:NY; 1:NZ]
         := BUFF
     (c) NR := NR + 1
   end do,
```

where "send" and "receive" are the message-passing sequences for transmitting data to another processor and that for receiving from other processors, respectively. The code has been written in Fortran, and the interprocessor communications is achieved using a P4 library (Butler and Lusk, 1994), which is a parallel programming utility similar to the PVM (Sunderam et al., 1994) and MPI (e.g., Gropp et al., 1995) environments. Since P4 can be implemented on various platforms for parallel computing, such as distributed-memory or shared-memory multi-CPU workstations and a cluster of workstations connected by a computer network, the parallel program can be implemented easily onto many styles of parallel computers.

Theoretical parallel performance

To predict the speed-up rate of our parallel program using a large number of processors, we formulate a simple performance prediction model defined by the ratio of total communication time to the other computation time.

We assume that the whole domain is a regular array of N^3 grid points partitioned among NP processors. The number of grid points per processor is then N^3/NP. We assume further that the computation is performed entirely in single-precision arithmetic (4 bytes per data) and that the derivatives of 18 field quantities (six for z-differentiation and 12 for horizontal differentiation) in the equation are processed with a rate of v_d bytes/s. Then, for each processor, the total computation time $T_t(NP)$ taken per iteration is

$$T_t(NP) = \frac{1}{\alpha} \times \frac{\dfrac{4 \times N^3}{NP} \times 18}{v_d}, \tag{12}$$

where α denotes the ratio of the time required for differentiation operation to the total computation time, which depends on the performance of the FFT program. We measured $\alpha \simeq 0.3$ for the real FFT.

We also assume that the parallel computer has an idealized architecture where the communication with every processor has the same communication speed, v_c bytes/s, regardless of the location of processor pairs, and the network has enough bandwidth to pass all the data without dropping the speed. Then,

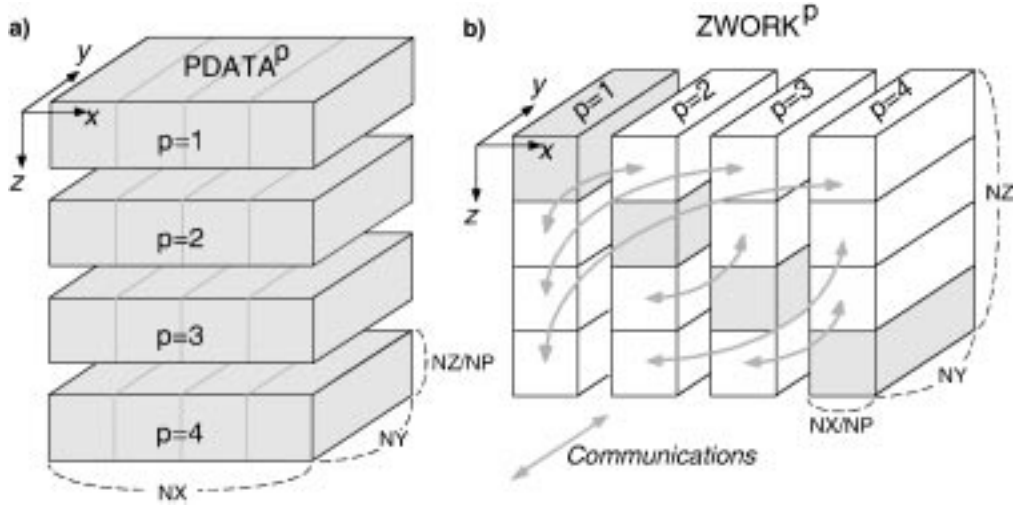

FIG. 1. Example of domain partition of the 3-D wavefield for a four-processors ($NP = 4$) model. (a) The volume is sliced horizontally into four pieces $PDATA^p$ ($p = 1, \ldots, 4$) of equal size and mapped onto four processors. The wavefield in each subdomain is calculated by each processor individually except for the z-differentiations. (b) Configuration of array data $ZWORK^p$ assigned to each processor for calculating the z-differentiation. Since each processor has only the part of data shaded in the figure in its subdomain, $PDATA^p$, and the other parts are on the different subdomains, communication is required to swap data with the other processors. After calculating the z-differentiation, the results are exchanged again between the other subdomains.

for each processor, the total computation time per iteration, including the time for the communications with $NP-1$ processors before and after calculating six z derivatives, is

$$\bar{T}_t(NP) = \frac{1}{\alpha}\cdot\frac{\dfrac{4\times N^3}{NP}\times 18}{v_d} + \frac{\dfrac{4\times N^3}{NP^2}\times 2\times (NP-1)\times 6}{v_c}. \tag{13}$$

Then, the efficiency of speed-up rate by NP processors, defined as $E_p(NP) = T_t(1)/\bar{T}_t(NP)$, is given by

$$E_p(NP) = \frac{NP}{1+\dfrac{2\alpha}{3}\times\dfrac{NP-1}{NP}\times\dfrac{v_d}{v_c}}. \tag{14}$$

This equation demonstrates that the parallel theoretical efficiency using NP processors, as only a function of α and the ratio of v_d/v_c, is independent of the size of the problem.

Figure 2 shows the theoretical speed-up rate for NP processor execution on the idealized systems that have different values of v_d/v_c. This figure demonstrates that the machine with relatively fast communications v_c (small v_d/v_c) can achieve a higher speed-up rate than that for slow v_c (large v_d/v_c).

PARALLEL PSEUDOSPECTRAL COMPUTATION—BENCHMARK TEST

In this section, we present benchmark results for the partitioned algorithm implemented on three different parallel platforms: Thinking Machine Corporation CM-5 parallel computer, a cluster of workstations of Sun Microsystems SPARC-10, and Digital Equipment DEC-Alpha 7200/620 multi-CPU workstation (DEC-Alpha). The parallel performance of each platform was tested with model sizes of 64^3 and $128^2\times 64$ grid points with a total computer memory requirement of 54 and 204 Mbytes.

The CM-5 computer employed here is a 32-node machine; each processor has a peak computing rate of 4.2 Mflops with a local memory of 64 Mbytes (1 Gbyte in total) connected by an interprocessor communication line with a bandwidth of 5 Mbytes/s, which represents one of the current standard parallel computing architectural environments.

We also implemented the parallel code on the cluster of eight SPARC-10 workstations connected by an Ethernet network. The machine has a 25 Mflops computational rate with various memory sizes between 32 and 128 Mbytes and a network bandwidth of 1.25 Mbytes/s. Such parallel computing by a workstation cluster has recently attracted considerable attention as an economical approach for high-performance computing. The architectural limitation of the workstation cluster for fast parallel computing is that the bandwidth of the computer network is much slower than that of the interprocessor communication line for the purpose-built parallel computers.

We experimented further with our parallel algorithm by implementing it on the DEC-Alpha workstation that has two processors of 200 Mflops computational rate and a shared memory of 2 Gbytes. Usually for the shared-memory machine, the parallel code generated automatically by the Fortran 90 compiler is often used, but we also experimented with the feasibility of our parallel code on this style of parallel environment.

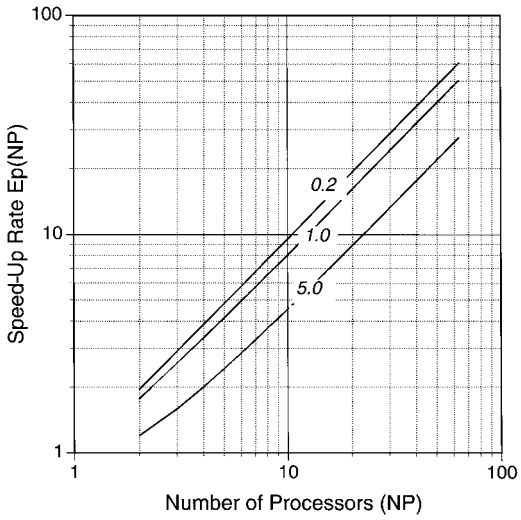

FIG. 2. Theoretical parallel efficiency of speed-up rate E_p by a number of processors (NP). The efficiency curves of different parallel computing architectural environments: $v_d/v_c = 0.2$, 1.0, and 5.0 are displayed.

Communication and differentiation speed

Table 1 shows the differentiation speed v_d of the processors of each platform, which was evaluated by measuring the time taken for calculating various sizes of derivatives by use of the real FFT on a single processor. The results of the experiments are consistent with the theoretical computational rates of the platforms.

Table 2 shows the interprocessor communication speed v_c of each parallel platform, which is measured by using a simple program to swap data packets between two processors of various data length. The result is almost consistent with the previous knowledge of the network speed of each system. The

Table 1. Differentiation speed v_d in Kbytes/s of each computer. The speed is measured using the real FFT with a different data length (bytes).

Data size, bytes	CM-5	DEC-Alpha	SPARC-10
256	420	4110	1470
512	400	4240	1520
1024	360	4000	1430
2048	330	3610	1290
4096	300	3200	1140

Table 2. Communication speed v_c in Kbytes/s of each computer, taken by swapping data between two processors using a different data packet size (bytes).

Data size, bytes	CM-5	DEC-Alpha	SPARC-10
256	840	2990	160
512	1200	5140	260
1024	2200	9700	320
2048	2800	14030	410
4096	3100	14600	420

communication between processors using the P4 system is defined by the linear function of two components—latency time as a fixed startup time and transmission time proportional to data length. Usually, the former is much longer than the latter (see Butler and Lusk, 1994), and therefore, v_c is relatively slow when using small data packets. It should be noted also that the actual communication speeds $\bar{v}_c(=\gamma \cdot v_c)$ in practical parallel computing using a large number of processors may decrease from the values shown in Table 2, because of the collision of data packets transmitting between many processors. We found that in an ideal parallel platform, such as the CM-5, having a very fast network line shows $\gamma \sim 1$ irrespective of the number of processors, while the workstation cluster connected by an Ethernet network usually shows $\gamma \sim 0.5$ when the load of the computer network is heavy. The deterioration of the parallel speed-up rate because of the collision of data transmission happens gradually as the number of processors increases and the performance of data transmission between processors decreases. To bring the parallel performance close to the theoretical speed-up rate, it is necessary to reconstruct the configuration of processors, especially when employing workstations on a busy computer network.

Benchmark results

We performed the benchmark test of the parallel pseudospectral calculations for small (64^3 grid points; Figure 3a) and medium ($128^2 \times 64$ grid points; Figure 3b) sized models. Since the models are too large to fit into the processor memory of the CM-5, the test was performed using more than four and eight processors for the small and large models, respectively.

Figure 3 illustrates the results of the benchmark test (lapsed time per iteration). The results demonstrate fairly good speed-up rates for both models as the number of processors increases. For example, the CM-5 achieves a maximum speed-up rate of 26 using 32 processors, and the SPARC-10 cluster shows 4.6 using eight workstations. Some measure of parallel gain is also found in the DEC-Alpha two-CPU machine that shows the maximum speed-up rate of 1.6 for the small model. Figure 3 also displays the theoretical speed-up curves calculated by equation (14) for the values of v_c and v_d of each system corresponding to the data size used in the modeling, which agrees well with the elapsed time of the experiments.

The speed-up rate for the CM-5, which is faster relative to the other machines, arises mainly because the v_d of the CM-5 is much slower than the v_c, and consequently, the ratio of v_d/v_c is much smaller ($0.16 \sim 0.33$) than the other machines. We find also that the SPARC-10 cluster that has a large v_d/v_c value ($4.5 \sim 5.8$) shows a relatively gentle speed-up rate, especially when using just a few workstations. For the DEC-Alpha, we performed the parallel computing with only two processors. Nevertheless, our parallel algorithm also demonstrates good performance on the shared-memory multi-CPU computer.

EXAMPLE

To show the feasibility of the parallel pseudospectral code for the fast computation using current parallel computing environments, we undertake 3-D modeling of seismic wave propagation for cases similar to those demonstrated in the previous studies using supercomputers.

Fault-zone trapped waves: Acoustic case

We consider an acoustic wave propagation trapped within a vertical low-velocity fault zone, which was also simulated by Huang et al. (1995). The model consists of $128^2 \times 64$ grid points with grid intervals of 0.02 km, in which a typical fault zone with width of 0.4 km and a velocity of 2.0 km/s is sandwiched between two quarter spaces with a velocity of 2.6 km/s. In the center of the fault zone, a point pressure source is placed at a

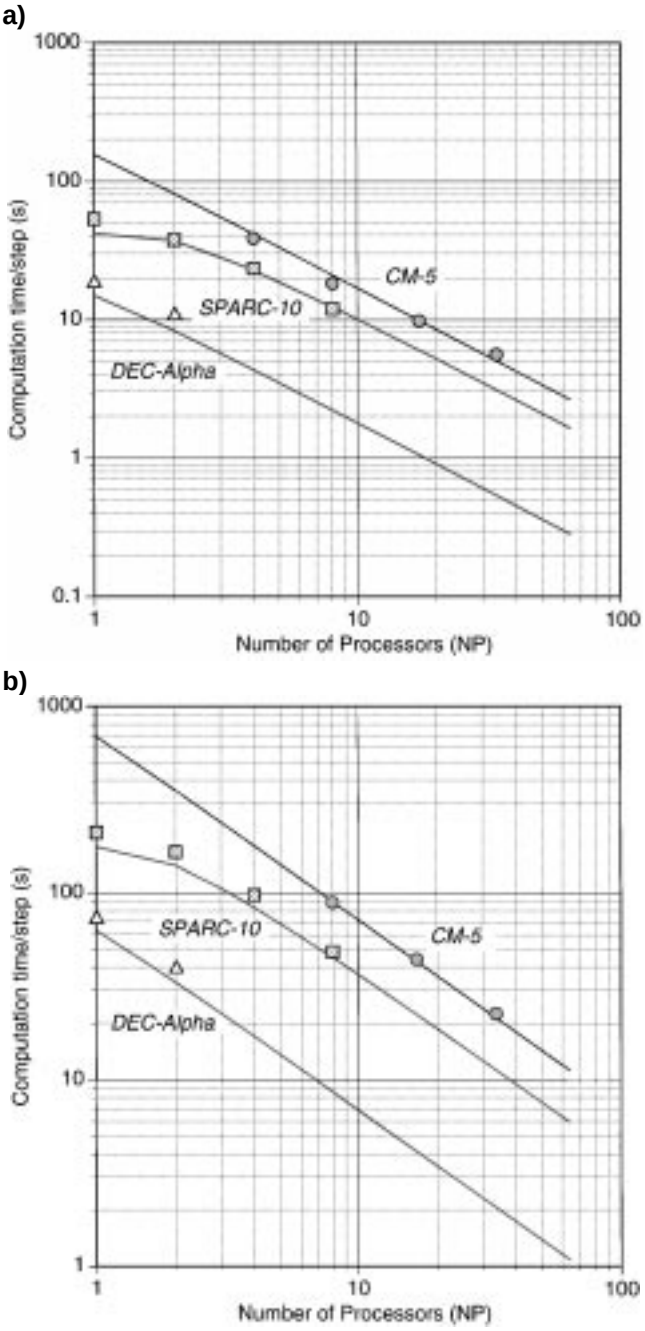

FIG. 3. Theoretical and experimental efficiency of the parallel computing measured by use of three different parallel environments with a number of processors: (a) for small (64^3) grid size model test, and (b) for medium ($128^2 \times 64$) grid size model.

depth of 0.21 km below the free surface, which has a source time function of a pseudodelta function with a maximum frequency of around 30 Hz. The density is assumed to be constant ($\rho = 2.0\,\mathrm{g/cm^3}$) for the entire model so that we can use the wave equation rather than the equation of motion to reduce the computation time and memory [see Huang et al., 1995; equation (2)]. The computation for this model requires a memory of 28 Mbytes and a computation time of 52 minutes to evaluate the wavefield of 460 steps using a cluster of eight SPARC-10 workstations. The computer performance of the SPARC-10 cluster is about 0.17 of that of the CRAY Y-MP supercomputer used by Huang et al. (1995).

Figure 4 displays the snapshots of the wavefield at different time steps, and Figure 5 shows the synthetic seismograms on a linear array (a—a' in Figure 4) at depth of 0.01 km (half of the grid size below the free surface). Each trace is normalized by its maximum amplitude to increase the visibility of the phases. In the snapshots, we see the wave forming a circular wavefront ($T = 0.2$ s frame) at the source, which is then propagating within the fault zone as multiple reflections (see $T = 0.4$ s and 0.6 s frame), with some portion of energy leaking out of the fault zone. Such a trapped wave can also be discerned as a remarkable later phase in the synthetic seismograms (Figure 5).

Fault-zone trapped waves: Elastic case

We now consider extending the modeling of a trapped wave in a fault zone to the realistic case using the partitioned pseudospectral calculation. The configuration of the model employed here is the same as that in the previous acoustic case. An explosion source is placed inside the fault zone with a *P*-wave velocity of 3.6 km/s, *S*-wave velocity of 2.0 km/s, and a density of $2.0\,\mathrm{g/cm^3}$ that is embedded in an otherwise homogeneous medium with a *P*-wave velocity of 4.6 km/s, *S*-wave velocity of 2.6 km/s, and a density of $2.2\,\mathrm{g/cm^3}$. This modeling requires a total memory of 204 Mbytes and a computation time of 7.5 hours using a 32-processor CM-5 to evaluate the wavefield for 1200 time steps.

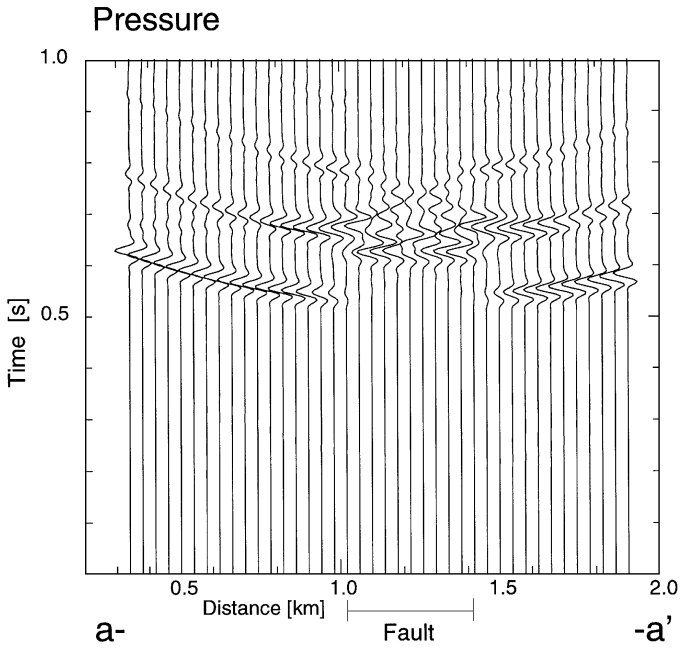

FIG. 5. Synthetic seismograms of pressure time section of the linear-array stations (labeled a—a' in Figure 4). Each trace is normalized by its maximum amplitude.

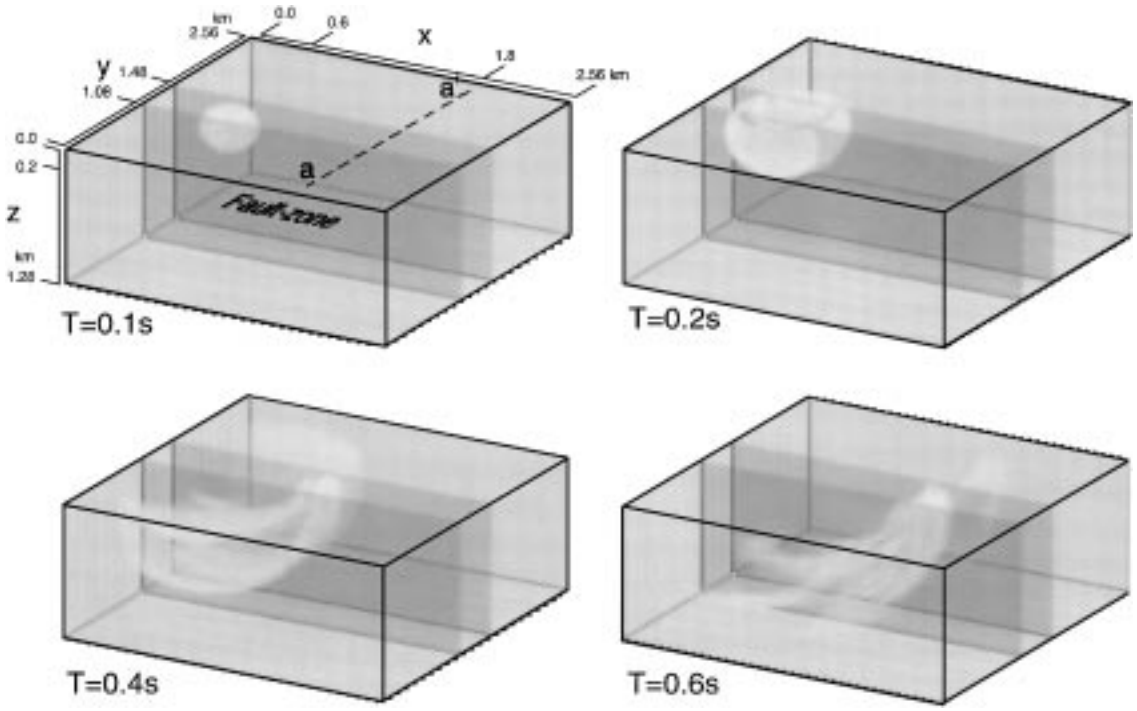

FIG. 4. Snapshots of the acoustic wavefield at different time steps for a fault-zone model. A pressure source is placed inside the fault zone with a depth of 0.2 km. The dashed line marked by a—a' denotes a linear array of the receivers.

Figure 6 displays the snapshots of the wavefield separated into *P*- and *S*-wave contributions by calculating the divergence and rotation of the wavefield, respectively. Figure 7 shows the synthetic three-component seismograms of the displacement along the linear array a—a' (see Figure 6). Each trace in the seismograms is normalized by its maximum amplitude to increase the visibility of phases. In Figure 7 we find the *P*-wave trapped within the fault zone, which clearly can be seen in the seismograms of vertical and *x*-components. Also, we can see *P* to *S* conversion caused at the free surface and the edge of the fault and propagating outside the fault zone, which is discerned clearly in the seismograms of both horizontal components. This modeling demonstrates that it may be possible to estimate the properties of the fault zone from the *P* to *S* conversions in the horizontal components seismograms.

Seismic reflection profiling of syncline model

Next, we perform an alternative modeling for the 3-D elastic wavefield, similar to that of Reshef et al.(1988b) (Figure 8). This model consists of a truncated dipping interface overlaying a syncline, where an explosion source with a 35-Hz-band-limited Ricker wavelet is placed 0.33 km below the free surface. The

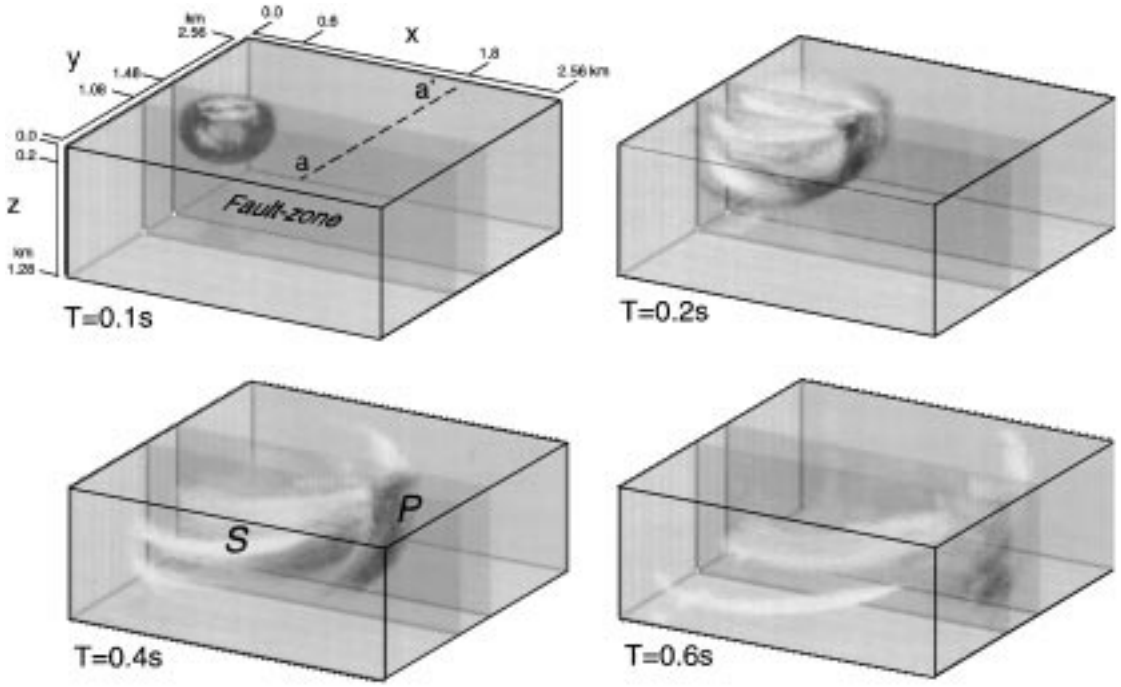

FIG. 6. Snapshots of elastic wavefield at different times for a fault-zone model; the wavefield is separated into *P*-wave (black) and *S*-wave (white) components.

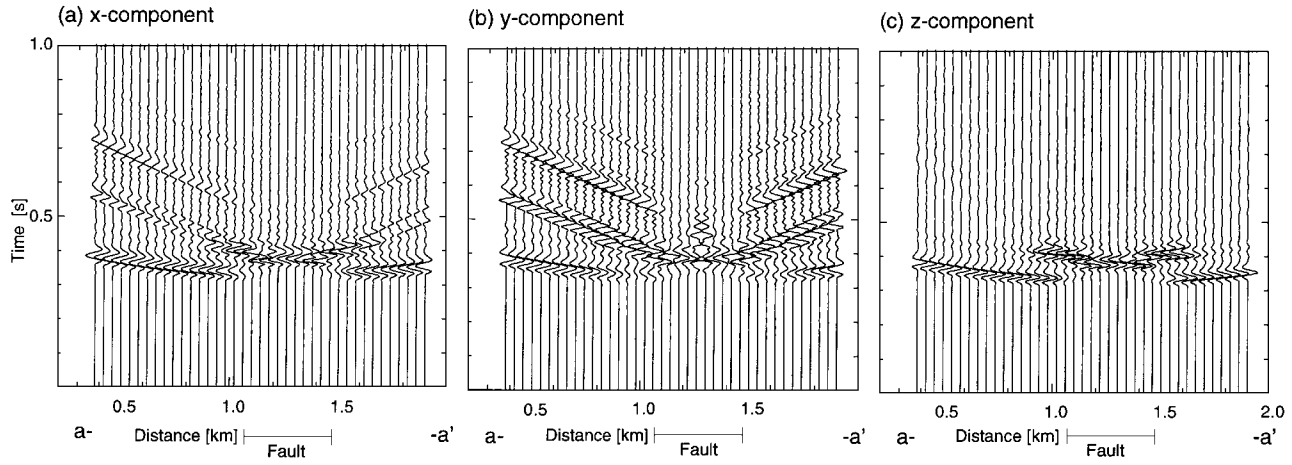

FIG. 7. Synthetic seismograms (three-components) at the stations for the fault-zone model. Each trace is normalized by its maximum amplitude.

model is discretized by spatial grid points of $256 \times 128 \times 256$ with a regular grid spacing of 0.02 km.

Figure 8 displays some snapshots of the wavefield at different time steps separated into *P*- and *S*-wave contributions, and Figure 9 shows the synthetic three-component seismograms of the stations (a—a' in Figure 8) placed along the *y*-direction crossing the source position. In the first two frames, we see the *P*-wavefront produced at the explosion source propagating in the first layer. In the $T = 0.6$ s frame, we find the *P* to *S* conversion generated at the dipping interface. The main feature in the last few frames is the pattern of multiple reflections. Such *S*-wave reflections are recognized easily in the synthetic seismograms (Figure 9).

The modeling has been performed on the DEC-Alpha workstation with two processors, which took a memory of 736 Mbytes and a total computation time of 87 hours for evaluating the wavefield of 1500 time steps. The performance of parallel pseudospectral computing by the two-CPU DEC-Alpha workstation is about 0.052 of the CRAY X-MP/48 four-CPU supercomputer system used in Reshef et al (1988b). This experiment may indicate the arrival of the 3-D elastic modeling age using parallel pseudospectral computing by using a multi-CPU workstation.

DISCUSSION AND CONCLUSION

3-D elastic modeling has long been an expensive application to realistic problems. The trend of steadily increasing computer power is bringing us closer to the use of 3-D elastic modeling for practical applications. For example, using a 2-Gbyte memory (a typical maximum memory size of a conventional UNIX operation system), the pseudospectral modeling can treat the elastic wavefield within a 128-wavelength cube. We have addressed a new parallel pseudospectral computing technique for the 3-D wavefield that can be implemented on many styles of current parallel platforms. The results of the benchmark test with various processors indicate that fairly good speed up can be achieved by using both high-performance parallel computers and an economical parallel platform of a workstation-cluster connecting network, that agree well with the theoretical performance prediction using the knowledge of computation-to-communication speed ratio of each system.

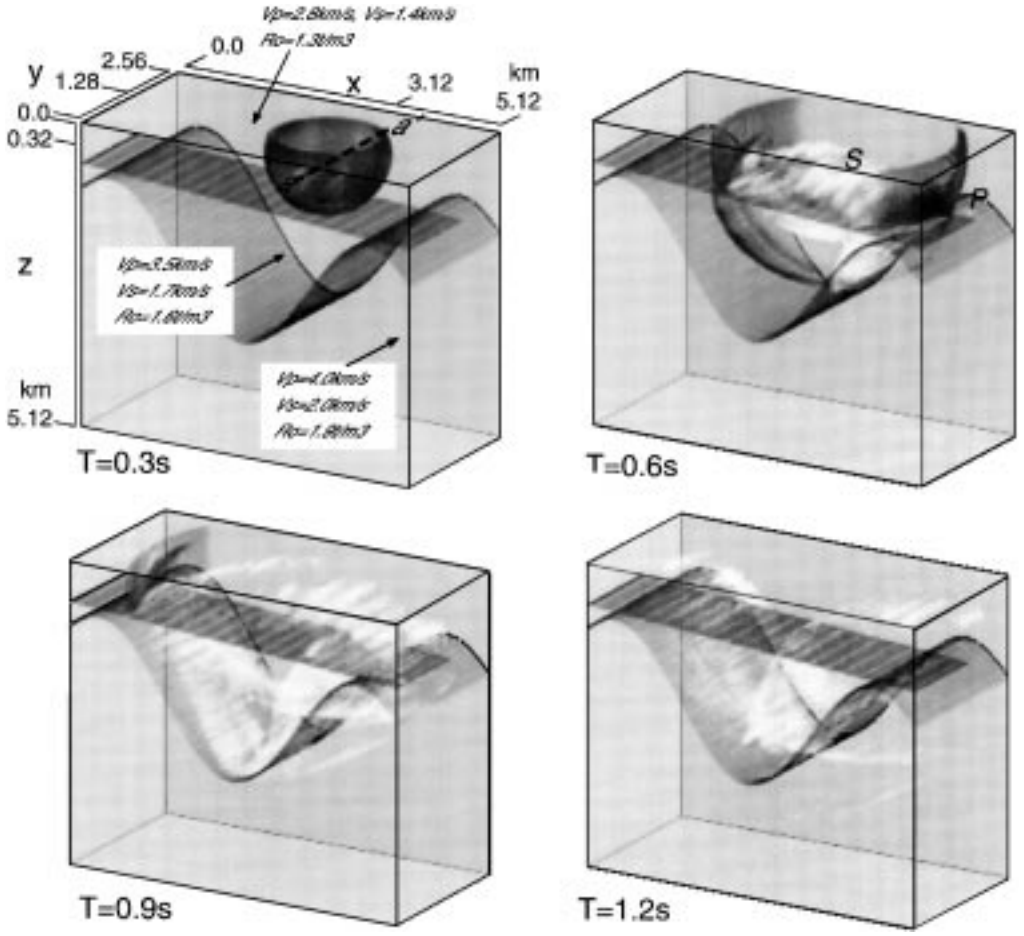

FIG. 8. Snapshots of the seismic wavefield for a syncline model with a dipping interface. The wavefield is separated into *P*-wave (black) and *S*-wave (white) contributions. The dashed line along the *y*-direction passing through the source position denotes the receiver locations.

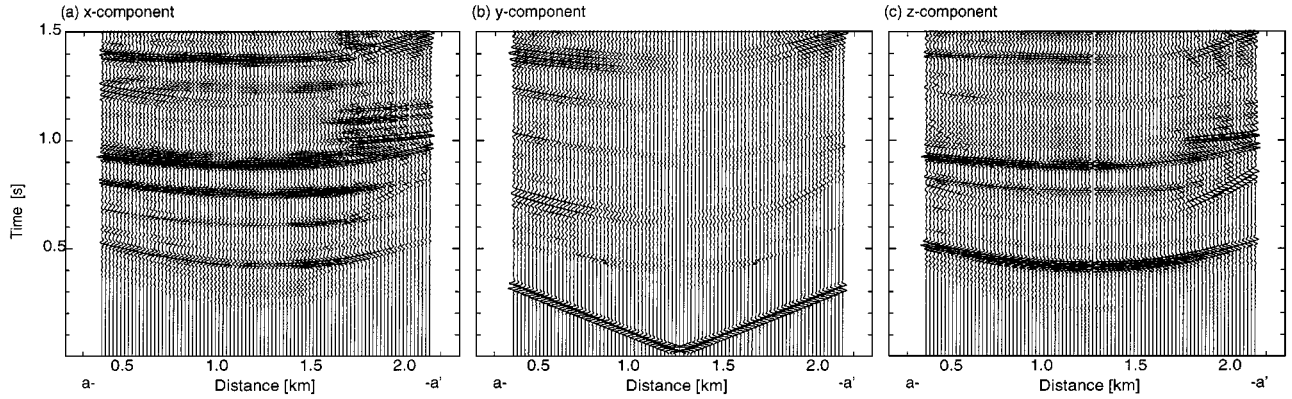

FIG. 9. Synthetic three-component seismograms at the receivers. Automatic gain control has been applied to enhance the amplitude of reflections.

ACKNOWLEDGMENTS

The authors thank the Australian National University Supercomputing Facility (ANUSF) for providing a CPU time grant on the CM-5, and the Hokkaido University of Education Information Processing Center for permission to use the DEC 7000/620 workstation. I would like to acknowledge Ichizo Ninomiya who kindly sent me his Fortran FFT program that I used in my simulation program. This FFT program is a part of NUMPAC and was developed by Ninomiya and his co-researchers (Ninomiya and Hadano, 1985). The image of seismic wavefield snapshots displayed in the paper was generated using the Stanford VolPack volume renderer developed by Lacroute and Levoy (1994). T. Furumura wishes to express gratitude to the Australian Academy of Science and the Japan Society for the Promotion of Science for a Scientist Exchange Program between the two countries. T. Furumura also gratefully acknowledges the support of the Research School of Earth Sciences, Australian National University, with a Visiting Fellowship during the period that this work was carried out. Constructive reviews by two anonymous reviewers are appreciated.

REFERENCES

Bergland, G. D., 1968, A fast Fourier transform algorithm for real-valued series: Comm. ACM,**11**, 703–710.

Butler, R. M., and Lusk., E. L., 1994, Monitors, messages, and clusters: The p4 parallel programming system: Parallel Comput., **20**, 547–564.

Cerjan, C., Kosloff, D., Kosloff, R., and Reshef, M., 1985, A nonreflecting boundary condition for discrete acoustic and elastic wave equations: Geophysics, **50**,705–708.

Chen, H.-W., and McMechan, G. A., 1993, 3-D physical modeling and pseudospectral simulation of seismic common-source data volumes: Geophysics, **58**, 121–133.

Daudt, C. R., Braile, L. W., Nowack, R. N., and Chiang, C. S., 1989, A comparison of finite-difference and Fourier method calculations of synthetic seismograms: Bull. Seis. Soc. Am.,**79**, 1210–1230.

Fornberg, B., 1987, The pseudospectral method: Comparisons with finite difference for the elastic wave equation: Geophysics, **52**, 483–501.

Furumura, T., and Takenaka, H., 1992, A stable method for numerical differentiation of data with discontinuities at end-points by means of Fourier transform-symmetric differentiation: Geophys. Expl., **45**, 303–309 (in Japanese with English abstract).

———1995, A wraparound elimination technique for the pseudospectral wave synthesis using an antiperiodic extension of the wavefield: Geophysics, **60**, 302–307.

———1996, 2.5-D modeling of elastic waves using the pseudospectral method: Geophys. J. Int., **124**, 820–832.

Furumura, T., Takenaka, H., and Ninomiya. I., 1993, Is the fast Hartley transform more efficient than FFT?: Trans. Japan Soc. Indust. Appl. Math., **3**, 245–255 (in Japanese with English abstract).

Gropp, W., Lusk, E., and Skjellum, A., 1995, Using MPI: The MIT Press.

Huang, B. S., Teng, T. L., and Yeh, Y. -T., 1995, Numerical modeling of fault-zone trapped waves: Acoustic case: Bull. Seis. Soc. Am., **85**, 1711–1717.

Kang, I. B., and McMechan, G. A., 1990, Two-dimensional elastic pseudospectral modeling of wide-aperture seismic array data with application to the Wichita Uplift-Anadarako Basin region of southwestern Oklahoma: Bull. Seis. Soc. Am., **80**, 1677–1695.

Kosloff, D., and Baysal, E., 1982, Forward modeling by a Fourier method: Geophysics, **47**, 1402–1412.

Kosloff, R., and Kosloff, D., 1986, Absorbing boundaries for wave propagation problems: J. Comput. Phy. **63**, 363–376.

Lacroute, P., and Levoy, M., 1994, Fast volume rendering using a shear-wrap factorization on the viewing transformation: ACM SIGGRAPH, **94**, 451–458.

Liao, Q., and McMechan, G. A., 1993, 2-D pseudospectral viscoacoustic modeling in a distributed-memory multiprocessor computer: Bull. Seis. Soc. Am., **83**, 1345–1354.

Ninomiya, I., and Hadano,Y., 1985, Numerical computing library NUMPAC: J. Info. Processing, **26**, 293–300 (in Japanese with English abstract)

Renaut, R. A., and Woo, M. L., 1992, Parallel pseudospectral method for the solution of the wave equation, *in* Wave propagation and inversion: Soc. Indus. Appl. Math.

Reshef, M., Kosloff, D., Edwards, M., and Hsiung, C., 1988a, Three-dimensional acoustic modeling by the Fourier method: Geophysics, **53**, 1175–1183.

———1988b, Three-dimensional elastic modeling by the Fourier method: Geophysics, **53**, 1184–1193.

Saatcilar, R., and Ergintav, S., 1991, Solving elastic wave equations with the Hartley method: Geophysics, **56**, 274–278.

Sato, T., Matsuoka, T., and Tsuru, T., 1994, Wavefield modeling by pseudospectral method on a parallel computer, Proc. Soc. Expl. Geophys. Japan conf., **90**, 96–99 (Japanese).

———1995, Wavefield modeling by pseudospectral method on a parallel computer(2)—Seismic data simulation and processing: Proc. Soc. Expl. Geophys. Japan Conf., **92**, 392–396 (Japanese).

Sorensen, H. V., Jones, D. J., Heideman, M. T., and Burrus, C. S., 1987, Real-valued fast Fourier transform algorithms: IEEE Trans. ASSP, **ASSP-35**, 849–863.

Sunderam, V. S., Geist, G. A., Dongarra, J., and Manchek, R., 1994, The PVM concurrent computing system: Evolution, experiences, and trends: Parallel Comput., **20**, 531–545.

Zhou, B., 1992, On: The use of the Hartley transform in geophysical applications, by R. Saatcilar, S. Ergintav, and N. Canitez (Geophysics, **55**, 1488–1495) and Solving elastic wave equation with the Hartley method, by R. Saatcilar and S. Ergintav (Geophysics, **56**, 274–278): Geophysics, **57**, 196–197.