ELSEVIER

Comput. Methods Appl. Mech. Engrg. 152 (1998) 85–102

Computer methods in applied mechanics and engineering

Large-scale simulation of elastic wave propagation in heterogeneous media on parallel computers

Hesheng Bao[a], Jacobo Bielak[a,*], Omar Ghattas[a], Loukas F. Kallivokas[a], David R. O'Hallaron[b], Jonathan R. Shewchuk[b], Jifeng Xu[a]

[a]*Computational Mechanics Laboratory, Dept. of Civil and Environmental Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA*

[b]*School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA*

Received 17 January 1997

Abstract

This paper reports on the development of a parallel numerical methodology for simulating large-scale earthquake-induced ground motion in highly heterogeneous basins. We target large sedimentary basins with contrasts in wavelengths of over an order of magnitude. Regular grid methods prove intractable for such problems. We overcome the problem of multiple physical scales by using unstructured finite elements on locally-resolved Delaunay triangulations derived from octree-based grids. The extremely large mesh sizes require special mesh generation techniques. Despite the method's multiresolution capability, large problem sizes necessitate the use of distributed memory parallel supercomputers to solve the elastic wave propagation problem. We have developed a system that helps automate the task of writing efficient portable unstructured mesh solvers for distributed memory parallel supercomputers. The numerical methodology and software system have been used to simulate the seismic response of the San Fernando Valley in Southern California to an aftershock of the 1994 Northridge Earthquake. We report on parallel performance on the Cray T3D for several models of the basin ranging in size from 35 000 to 77 million tetrahedra. The results indicate that, despite the highly irregular structure of the problem, excellent performance and scalability are achieved.

1. Introduction

The reduction of earthquake risk to the general population is a major problem facing countries located in highly seismic regions. Assessing the free-field ground motion to which a structure will be exposed during its lifetime is a critical first step for the design of new structures and retrofit of existing ones.

The main objective of this paper is to describe a methodology for modeling ground motion on parallel computers in large sediment-filled basins during earthquakes, and to illustrate with an application to a real basin. Observations during recent strong earthquakes have shown that three-dimensional local site effects, caused by the waves generated inside a basin, can result in strong ground motion amplification and longer duration of the surface ground motion, with respect to that in rock, and a rapid spatial variation of the ground motion that can cause large differential base motion of extended structures such as bridges or dams [19,10,3]. See [1] for a general overview, and [12,11,24,8,9,18,16], for instance, for representative recent work in this field.

Simulating the earthquake response of a large basin is accomplished by solving numerically the partial differential equations of elastic wave propagation, i.e. the Navier equations of elastodynamics. Several numerical methods have been used for approximating the solution to these problems. Boundary element methods

* Corresponding author.

PII S0045-7825(97)00183-7

have been popular for recovering solutions primarily in the frequency domain and for moderately-sized linear problems. Inhomogeneities, nonlinearities, the large scale of such basins as Los Angeles and Kanto, and the desire to model directly in the time domain, preclude their use here. On the other hand, uniform grid domain methods such as structured finite differences become impractical for the very large problem sizes involved.

To see why uniform grids are impractical, consider the Los Angeles Basin. For a shear-wave velocity of 0.4 km/s and a frequency of 2 Hz, a regular discretization of the elasticity operator would place grid points 0.02 km apart to achieve second order accuracy. The region of interest has dimensions 140 km × 100 km × 20 km; thus, a regular discretization, governed by the softest layer, requires 35 billion grid points with three displacement components per grid point. At least a terabyte of primary memory would be needed, and on the order of 10^{13} operations would be required at each time step. The stability condition associated with explicit time integration of the semidiscrete equations of motion imposes a time increment at least as small as 0.004 s. Thus, a computer would have to perform at a sustained teraflop per second for two days to simulate a minute of shaking.

Instead, we use unstructured mesh finite element methods that tailor the mesh size to the local wavelength of propagating waves [2]. For a basin such as Los Angeles, the shear-wave velocity varies from 220 m/s to 4500 m/s throughout the basin and its vicinity. Since in three dimensions mesh density varies with the cube of shear-wave velocity, and since the softest soils are concentrated in the top layers, this means that an unstructured mesh method may yield three orders of magnitude fewer equations than with structured grids. Modeling the Los Angeles Basin for values of earthquake period and wave velocity that are desirable for engineering purposes thus becomes practical on the largest of today's parallel supercomputers.

We favor finite element methods for their ability to efficiently resolve multiscale phenomena, the ease with which they handle traction boundary conditions, and their firm theoretical foundation. For temporal approximation, we have studied both explicit and preconditioned conjugate gradient-based implicit methods. For hyperbolic problems, explicit methods become unstable if the time step is greater than the time it takes an elastic wave to cross any element—the Courant condition. Unconditional stability, on the other hand, can be achieved if one uses implicit methods. This implies that larger time steps can be taken. However, the very characteristic that makes them stable—the fact that the solution at a node at time $t + \Delta t$ requires information from all nodes at time t as opposed to just the neighbors—renders them unattractive on distributed memory computers, since this implies global information exchange. One approach to making implicit methods efficient on parallel computers is to use iterative methods for their solution, effectively rendering them explicit. However, we have found that our mesh generators give us such good control over mesh resolution that the Courant condition for explicit methods is not onerous. The result is that the more readily parallelizable explicit methods perform better for elastic wave propagation problems. In this paper we consider only a single-step explicit time integration method.

While unstructured mesh methods for simulating wave propagation through heterogeneous media result in many fewer equations, they introduce a number of computational difficulties that must be overcome. First, mesh resolution must closely follow wavelength; too coarse a resolution will introduce error, too fine will result in unnecessary computation as well as excessively small time steps (when explicit integration methods are used). Second, element aspect ratios must remain small; large aspect ratios will eventually result in instability in the time integration scheme. Highly heterogeneous basins, in which wavelengths vary rapidly in space, introduce special difficulties when trying to follow the wavelength change without severely stretching the mesh. Third, unstructured mesh methods are not easy to program on parallel computers; their irregular data structures require nontrivial mappings onto parallel machines and irregular communication patterns are generated. Thus, we have developed fast, robust computational geometry and mesh generation techniques for highly spatially-variable meshes, and compilers and tools that simplify the programming of unstructured mesh methods on parallel systems.

For an alternative approach to parallel ground motion modeling on distributed memory machines, see e.g. the work of Olsen et al. [17], which employs finite differences on regular grids. See also the references to prior finite difference modeling work on sequential machines contained therein. In addition to the finite element method described in this paper, there have been recent efforts to endow finite difference wave propagation methods with multi-resolution capabilities. See the work described in [14], which uses composites of regular grids to achieve variable resolution.

In the remainder of this paper, we present numerical methods and geometric algorithms for modeling earthquake-induced ground motion in highly heterogeneous basins. We also describe an automatic code generator for solution of unstructured mesh PDE problems on parallel distributed memory computers. As an

application, we model the response of the San Fernando Valley in Southern California to an aftershock of the 1994 Northridge earthquake, and give performance results on the Cray T3D.

2. Methodology

In this section we describe spatial and temporal discretization and solution techniques for the governing elastodynamics equations, which are performed in parallel. We also describe the sequential phase, which consists of a mesh generator capable of resolving local wavelengths, a mesh partitioner that rapidly provides asymptotically optimal partitions, several initialization steps that are carried out prior to parallel solution of the discrete wave propagation equations, and the parallel code generator, which shields the user from issues of parallelism and communication, while allowing a high-level description of the numerical method.

2.1. Governing equations and discretization

This section presents the parallel numerical techniques we use to approximate the solution of the wave propagation equations. Navier's equations of elastodynamics for an isotropic, heterogeneous medium are

$$\nabla \cdot [\mu(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^{\mathrm{T}}) + \lambda(\nabla \cdot \boldsymbol{u})\boldsymbol{I}] = \rho \frac{\partial^2 \boldsymbol{u}}{\partial t^2}, \tag{1}$$

where $\boldsymbol{u}$ is the displacement vector field, ρ is the density, and λ and μ are elastic material constants, which depend on the shear (c_s) and dilatational (c_p) wave velocities according to

$$c_s = \sqrt{\frac{\mu}{\rho}}, \qquad c_p = \sqrt{\frac{\lambda + 2\mu}{\rho}}. \tag{2}$$

The domain of the problem is semi-infinite, yet our computational domain must be rendered finite. Here, we chose a rectangular parallelepiped whose size is determined by the extent of the valley. We prescribe absorbing boundary conditions that are local in both time and space on all sides of the computational domain except for the top surface, which is traction free. In this application, we use the simplest possible absorbing boundary—a damper. While a viscous damper is a sufficient choice for high frequencies and normal wave incidences, it is, in general, a poor approximant of the exact condition at the truncation interface. However, we choose the truncation surfaces so that they lie outside the sedimentary valley, that is, mostly in rock. In this way, we ensure that the amplitudes of the waves impinging upon the truncation boundaries will have been greatly reduced, thus minimizing spurious reflections.

Since, in many cases, the earthquake source can be outside the computational domain, its effect must be introduced into the region. This is carried out as described in [4,6] by means of effective forces. In short, for an arbitrary earthquake excitation these forces are determined in terms of the free-field motion by introducing a fictitious auxiliary surface that surrounds the basin. Across this auxiliary surface one imposes the conditions of continuity of displacement and traction. By selecting the total displacement vector field as the unknown in the resulting interior region and the scattered displacement field in the exterior region, the free-field displacement and traction now appear explicitly in the continuity conditions, which become jump conditions, with the free-field displacement and traction on the right-hand side. These non-homogeneous terms on the right-hand side are the ones that give rise to the effective forces upon spatial discretization. If, on the other hand, the seismic source is located inside the computational domain, say as a kinematic dislocation across a fault, one can select the fault itself as the auxiliary surface. The procedure is similar, but now one uses the total displacement everywhere as the unknown field; thus, the displacement field again experiences a jump across the interface, but the traction remains continuous. Notice that with this technique, whether the source is originally located inside or outside the computational domain, only outgoing waves will impinge upon the absorbing boundary. Both types of source are implemented in our code.

We also model material damping in the basin via viscous damping. With these modifications, standard Galerkin discretization in space by finite elements produces a system of ordinary differential equations of the form

$$\boldsymbol{M}\ddot{\boldsymbol{u}} + \boldsymbol{C}\dot{\boldsymbol{u}} + \boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}\,, \tag{3}$$

where $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{C}$ is the damping matrix associated with the absorbing boundary and material damping, $\boldsymbol{K}$ is the stiffness matrix, and $\boldsymbol{f}$ is the effective force vector. Here, $\boldsymbol{M}$, $\boldsymbol{C}$ and $\boldsymbol{K}$ are block matrices; the (i, j)th block of $\boldsymbol{M}$ is a 3×3 matrix given by

$$\boldsymbol{M}_{ij} = \int_{\Omega} \rho \phi_i \phi_j \boldsymbol{I}\, \mathrm{d}\Omega\,, \tag{4}$$

and the (i, j)th block of $\boldsymbol{K}$ is given by

$$\boldsymbol{K}_{ij} = \int_{\Omega} (\mu + \lambda) \nabla\phi_i \nabla\phi_j^{\mathrm{T}}\, \mathrm{d}\Omega + \int_{\Omega} \mu \nabla\phi_i^{\mathrm{T}} \nabla\phi_j \boldsymbol{I}\, \mathrm{d}\Omega\,, \tag{5}$$

where ϕ_i is the finite element global basis function associated with the ith node.

Damping is introduced through a Rayleigh damping approximation at the element level, i.e. we take

$$\boldsymbol{C}^e = \alpha \boldsymbol{M}^e + \beta \boldsymbol{K}^e\,, \tag{6}$$

where α and β are scalar constants and the superscript e indicates an element matrix. The first term leads to a damping factor that is inversely proportional to frequency, and the second to one that is linear in frequency. The constants α and β, which may vary within the basin according to the type of material, are chosen to best fit a prescribed attenuation law.

Given appropriate initial conditions, the system of ODEs (3) can be integrated in time using central differences, yielding the explicit method

$$\left(\boldsymbol{M} + \frac{\Delta t}{2}\boldsymbol{C}\right)\boldsymbol{u}_{t+\Delta t} = \Delta t^2 \boldsymbol{f}_t - (\Delta t^2 \boldsymbol{K} - 2\boldsymbol{M})\boldsymbol{u}_t - \left(\boldsymbol{M} - \frac{\Delta t}{2}\boldsymbol{C}\right)\boldsymbol{u}_{t-\Delta t}\,. \tag{7}$$

This method exhibits second-order accuracy in time; when coupled with linear finite elements, we obtain second-order accuracy in space as well. We use a lumped mass approximation to $\boldsymbol{M}$, which amounts to numerically integrating (4) with integration points at element vertices. This results in a diagonal mass matrix. To render the left-hand side operator of (7) diagonal, we further evaluate the off-diagonal components of $\boldsymbol{C}$ at $\boldsymbol{u}_t$ rather than $\boldsymbol{u}_{t+\Delta t}$. Inversion of the time stepping operator thus requires only a scaling of the right-hand side of (7), which is carried out just once prior to time stepping. Forming the products of $\boldsymbol{K}$ and $\boldsymbol{C}$ with vectors comprises the major computational effort associated with iterating on (7). The sparsity structure of $\boldsymbol{K}$ is dictated by the underlying finite element mesh, and is thus very irregular. If shear waves are not over-resolved, the time step necessitated by stability is of the order of the time step dictated by accuracy, which is what an implicit method would take. By choosing an explicit method we avoid solving linear systems at each time step. Thus, overall, the explicit method is superior for our application, especially on a parallel computer.

We have tried several different choices of basis function order, and have concluded that piecewise-linear functions are the most efficient for problems requiring engineering accuracy. Our conclusion is based on numerical experimentation using plane Ricker wavelets on unstructured homogeneous meshes (in which case we know what the exact solution should be), but a simple argument can be given as follows. We recognize first that (spatial) approximation errors are bounded from above by interpolation errors. We then ask, for a given order of basis function and a given acceptable level of infinity-norm error, how many nodal points are required to produce a piecewise-polynomial interpolant of a simple harmonic wave. Next, we convert the required number of nodes per wavelength to an estimate of the storage and work required for an iteration of the explicit method (7). For example, on a regular grid, if N is the total number of nodes, one can show that trilinear hexahedra require $163.5N$ words of storage and $498N$ flops/time step, while triquadratic hexahedra necessitate $357N$ words and $1164N$ flops/time step. So, for example triquadratic elements should require at least 2.2 times fewer nodes in order to be preferred (for storage reasons) over trilinear elements. However, one-dimensional interpolation suggests that 5% error requires 10 nodes per wavelength using linear elements or 9.4 nodes using quadratics. Thus, in three dimensions, triquadratics only allow $(10/9.4)^3 = 1.2$ times fewer nodes than trilinears, and are thus not warranted. An opposite conclusion is reached if one demands 99% accuracy. Our confidence in the values of material properties and in the fidelity of the source models for this problem does not warrant

solution accuracies greater than 95%. Thus, we conclude that for this level of accuracy, the powers of higher-order interpolation are offset by their increased cost, both in storage and in increased work.

2.2. Mesh generation

As we have seen, seismic wave propagation problems place special demands on mesh generators, including the need for tight control over mesh resolution and aspect ratio, and the need to support extremely large problem sizes. We have developed a fast, stable and efficient meshing algorithm for generating very large scale meshes, suitable for the large basins we target. Since repeated computations will be performed with a single mesh (one or two dozen earthquake scenarios, each involving thousands of time steps), we have decided to generate and partition each mesh sequentially. However, care must be taken in designing and implementing efficient algorithms for these steps, lest they become bottlenecks.

Mesh generation begins with a database of the material properties of the soil and rock within and around a particular basin. The material properties—the shear-wave velocity, the dilatational-wave velocity, and the density—are estimated throughout the basin from soil borings, from geological records, and from seismic prospecting studies.

The meshing algorithm comprises two steps. First, we generate an octree that resolves the local wavelength of shear waves. The wavelength is known from the shear-wave velocity and the frequency of excitation. We have seen in the previous section that 8–10 nodes per wavelength is sufficient for 'engineering', or 95%, accuracy when using linear finite elements. When constructing the octree, we enforce the rule that adjacent cells may not differ in edge length by more than a factor of two, producing a *balanced* octree. This is crucial for producing elements with bounded aspect ratios, since aspect ratios far from unity lead to poorly conditioned stiffness matrices, which, in turn, can lead to instability in time integration.

Once a balanced octree is created such that no cell is wider than one-tenth the length of the wave that passes through it, a finite element node is placed at each cell vertex. This set of nodes is then tetrahedralized according to the Delaunay criterion.[1] Delaunay tetrahedralization is performed by a straightforward implementation of the Bowyer/Watson incremental algorithm [5,25], which constructs the tetrahedralization by adding one node at a time and locally adjusting the mesh to maintain the Delaunay criterion.

We have found that the Bowyer/Watson algorithm is occasionally sensitive to floating-point roundoff error; tetrahedral mesh generation can fail dramatically because of roundoff when processing near-degenerate geometric features. Such failures became increasingly common for us as the size of our meshes grew. To overcome this problem, we have developed a method for fast exact arithmetic that is particularly well-suited for certain tests that arise in computational geometry codes [20]. Our method is used to construct predicates that determine whether a point falls to the left or right side of a line, or whether a point falls inside or outside a sphere. These predicates are adaptive in the sense that they only use exact arithmetic to the extent it is needed to ensure a correct answer. Hence, if a point falls very close to a line, high precision arithmetic may be needed to resolve which side of the line it falls on; if a point is far from a line, approximate arithmetic will suffice, so the test can be performed quickly. Because the latter case is far more common, our exact arithmetic predicates are on average only slightly slower than ordinary, non-robust floating-point predicates, and our Delaunay tetrahedralization code runs quickly while ensuring the integrity of its results.

Our use of the Delaunay tetrahedralization of the vertices of a balanced octree guarantees that element aspect ratios are bounded, and that element sizes are chosen appropriately so that wavelengths are sufficiently resolved without unnecessary resolution (provided the material properties do not vary too rapidly).

2.3. Mesh partitioning

Once a mesh is generated, the set of elements that comprise it must be partitioned into subdomains. Each subdomain can then be mapped onto a processor of a parallel machine. The goal of mesh partitioning is to

[1] We could have used a hexahedral mesh directly from the octree, but we would have had to introduce constraints at midside nodes to make the elements conforming.

minimize communication time while maintaining load balance. In an explicit method, communication is associated with the nodes that lie on the boundaries between subdomains and are shared by more than one processor. Processors sharing a node must communicate six words per shared node for each matrix–vector multiply, i.e. twice each time step in our method. Communication time depends on both the message sizes, which increase with the number of shared nodes, and the number of messages, which increases with the number of adjacent subdomains. The load on a processor for explicit solution of linear wave propagation problems is easy to predict: it is proportional to the number of nodes on that processor. Prediction becomes more difficult when nonlinearities are present, such as with the soil plasticity models that we are currently introducing into our code. In these cases, the work per node is solution-dependent. Nevertheless, for our purposes, we consider a mesh partitioner desirable if it produces subdomains of nearly equal size (where size is measured by number of elements and not by volume) and with as few nodes shared between processors as is reasonably possible.

The partitioner we use is based on the algorithm of Miller et al. [15]. This algorithm uses geometric information to construct a separator, i.e. a set of nodes whose removal separates the mesh into two pieces of roughly equal size. Each of these pieces is then recursively partitioned until the desired number of subdomains is reached. The Miller et al. algorithm produces separators that are asymptotically optimal; their length is of order $O(N^{2/3})$ in three dimensions, where N is the number of nodes. Theoretically, the algorithm runs in randomized linear time; in practice, the algorithm rapidly produces high quality partitions.

2.4. Parceling

After a mesh is partitioned into subdomains, there remain several operations that have to be performed on the partitions to prepare the input for the parallel program. We refer to these steps collectively as *parceling*. The steps include generating (i) the communication schedule for each processor; (ii) the global-to-local mapping information, which allows identification of a node or element number on a processor by its global number; and (iii) the nonzero structure of the stiffness matrix on each processor. The last item could be performed in parallel, but it takes little time and provides us with useful statistics on the mesh, so we perform it sequentially.

2.5. Code generation

The mesh generator, the mesh partitioner, and the parceler are all components of a general-purpose toolset for the efficient mapping of unstructured mesh computations arising from numerical solution of PDEs onto parallel systems. We refer to the toolset as the Archimedes system [7,22]; it is depicted in Fig. 1. Input to Archimedes includes (i) the problem geometry and material properties and (ii) a sequential program containing an element-level description of the finite element approximation, as well as a high-level description of the solution method.

The input program is written in a special-purpose C-like language augmented with finite element-specific and linear algebraic primitive operations that include element-level vector and matrix assembly, imposition of boundary conditions, sparse matrix–vector products, dot products, and pre-conditioning. Additional functions are specific to elastic wave propagation, and include absorbing boundaries, damping and seismic input incorporation. Archimedes programs contain no explicit communication statements, and thus can be written without any knowledge of the parallel machine's underlying communication system. The set of primitives that Archimedes understands is rich enough to express algorithms for solution of linear and nonlinear scalar and vector PDEs, using arbitrary-order finite elements in space and both explicit and implicit methods in time. For implicit methods, the Archimedes language provides for expression of various stationary iterative solvers as well as Krylov subspace methods. Furthermore, users can add new primitives as the need arises.

Once the input program is complete, Author, the code-generator component of Archimedes, creates parallel code. Archimedes' parallelizing compiler will generate code for any parallel system with C and MPI implementations, including networks of workstations (using the Argonne/Mississippi State MPICH implementation), Intel's Paragon (also using MPICH) and the Cray T3D (using the CRI/EPCC MPI implementation).

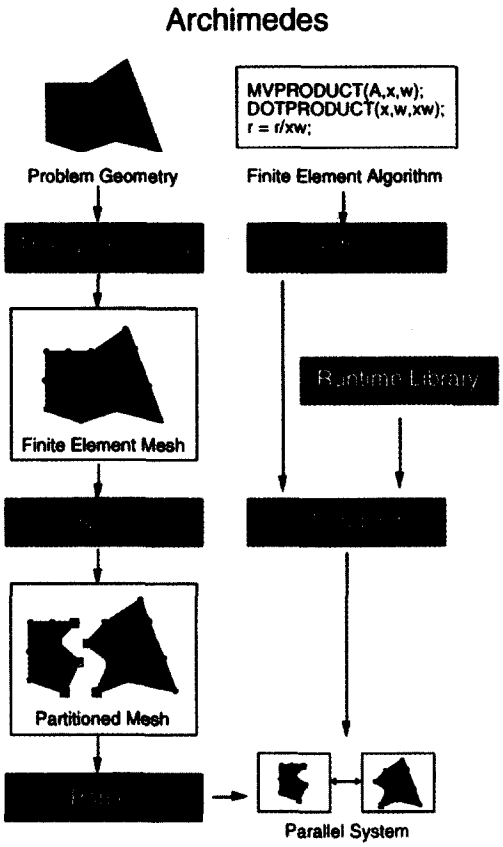

Fig. 1. The Archimedes system.

3. Application to the San Fernando Valley

We have applied the methodology outlined in the previous section to the modeling of earthquake-induced ground motion in the San Fernando Basin in Southern California. Specifically, the geographic region extending from -118.7500 to -118.1628 degrees longitude and -34.0828 to -34.3833 degrees latitude is modeled as a rectangular parallelepiped 54 km long by 33 km wide by 15 km deep. The properties of the basin are described by the triplet of shear-wave velocity, dilatational-wave velocity and density; for this application the material database has been obtained from the geology-based velocity model of Magistrale et al. [13]. For the above geographic region, Fig. 2 shows the shear-wave velocity distribution at a depth of one meter from the surface of the valley. The figure shows a variation in shear-wave velocity of at least a factor of seven. This factor exceeds 20 when properties at depth are considered.

3.1. Sequential phase

We use our mesh generator to create a mesh of the San Fernando Basin with a 220 m/s shear-wave velocity in the softest soil, assuming a seismic scenario with frequencies up to 1.6 Hz. Fig. 3 depicts the nodes generated by the balanced octree. The octree produces 13 million nodes, but many fewer are shown for clarity. As can be seen from the figure, the density of nodes is highest in the softest soil, where the shortest wavelengths occur. A regular grid for this material model would have resulted in 200 times the number of grid points. A Delaunay tetrahedralization of the set of 13 million nodes produces a mesh of 77 million tetrahedra. The mesh is generated in 13 h on one processor of a DEC 8400 and requires 7.7 Gb of memory. It has a maximum aspect ratio of 5.5 and exhibits a spatial resolution variability of over an order of magnitude. Fig. 4 shows the resulting mesh of tetrahedra, again coarsened for visualization purposes. Next, the 77 million element mesh is partitioned into 256

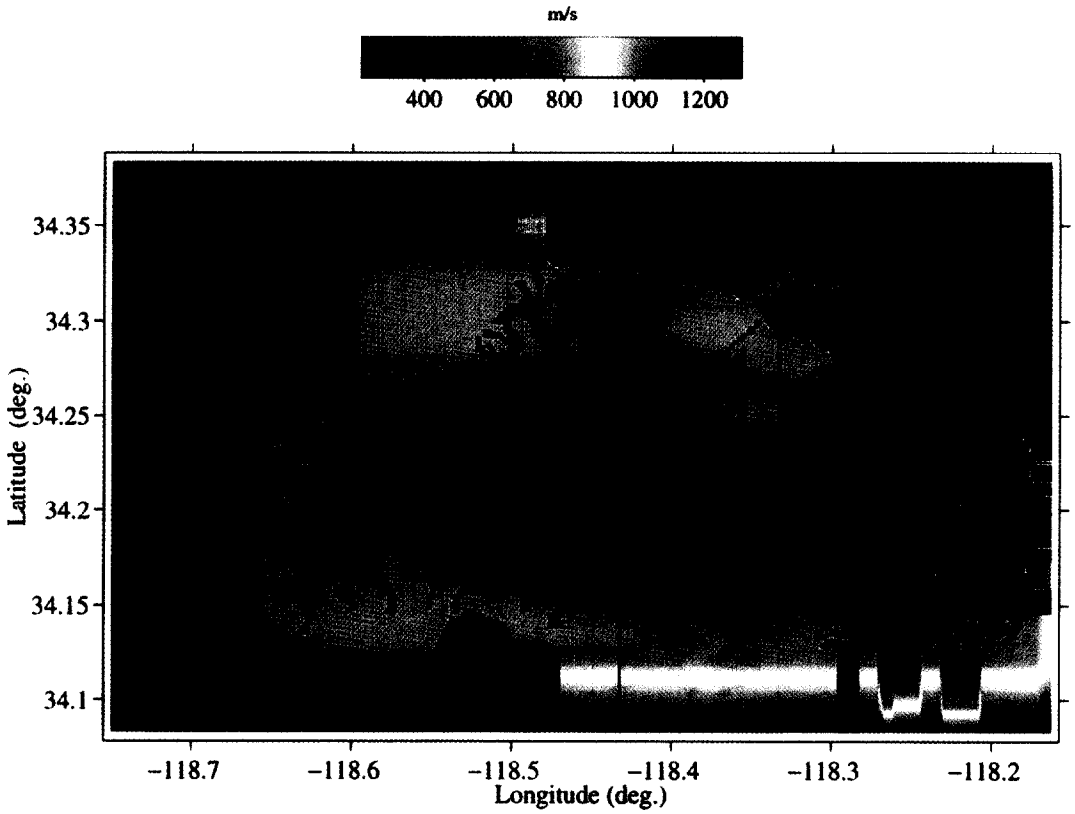

Fig. 2. Near-surface distribution of shear-wave velocity in the San Fernando Valley; actual depth is 1 m.

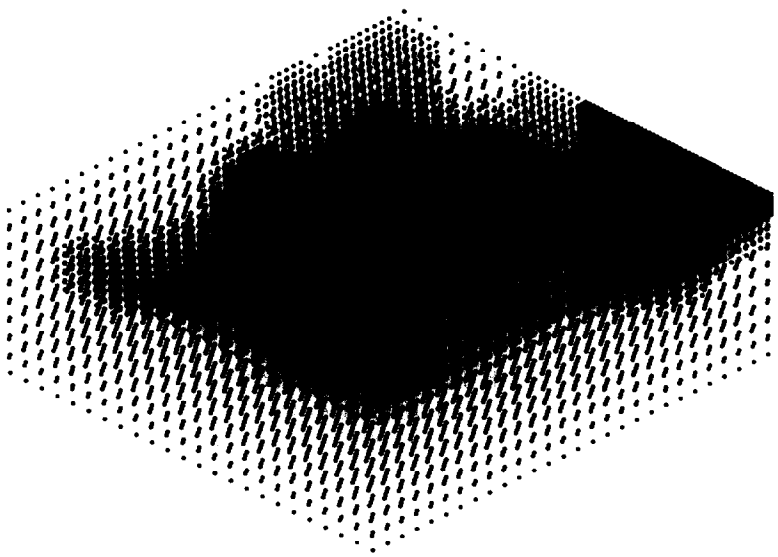

Fig. 3. Nodal distribution for the San Fernando Valley. Node generation is based on an octree method that locally resolves the elastic wavelength. The node distribution shown here is a factor of 12 coarser in each direction than the real one used for simulation, which is too fine to be shown, and appears solid black when displayed. However, the relative resolution between soft soil regions and rock illustrated here is similar to that of the 13 million node model we use for simulations.

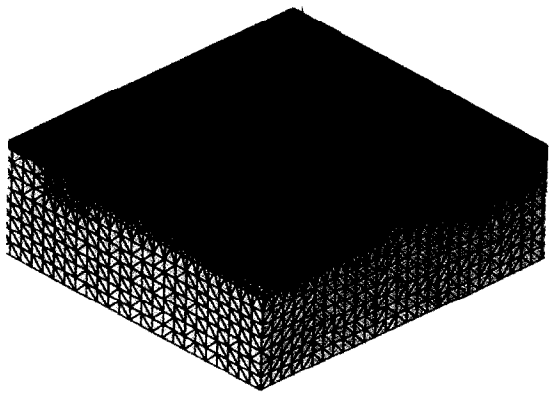

Fig. 4. Tetrahedral element mesh of the San Fernando Valley. Maximum tetrahedral aspect ratio is 5.5. Again, for illustration purposes, the mesh shown is much coarser than those used for simulation.

subdomains in about 3.8 h on one processor of the DEC 8400, and requires 7.9 Gb of memory. The resulting partition (for the coarser mesh and for 64 subdomains) is shown in Fig. 5. The figure shows the circular cuts characteristic of the partitioner. Despite the high spatial variability of the mesh, the partitions appear to be well-shaped.

The last step of the pre-simulation sequential phase is parceling, i.e. generating the communication schedule, the global-to-local mapping, and the global matrix nonzero structure. On the DEC 8400, parceling requires about 2.3 h and 7.7 Gb memory for the 77 million element San Fernando Basin mesh. The communication graph generated by the parceler is shown in Fig. 6. Each vertex represents a subdomain and corresponding processor; each edge represents communication between two processors.

3.2. Parallel phase

In this section we describe numerical results corresponding to the response of the San Fernando valley and provide timings that characterize the performance of the parallel explicit wave propagation code on the Cray T3D.

The San Fernando simulations involve meshes of up to 77 million tetrahedra and 40 million equations. As mentioned before, the largest mesh corresponds to a shear-wave velocity range of 220 m/s (softest layer) to 4500 m/s (rock) and a maximum frequency of 1.6 Hz; the code requires nearly 16 Gb of memory and takes 7.2 h (5.0 h excluding I/O) to execute for 16 667 times steps on 256 processors of the Cray T3D at the

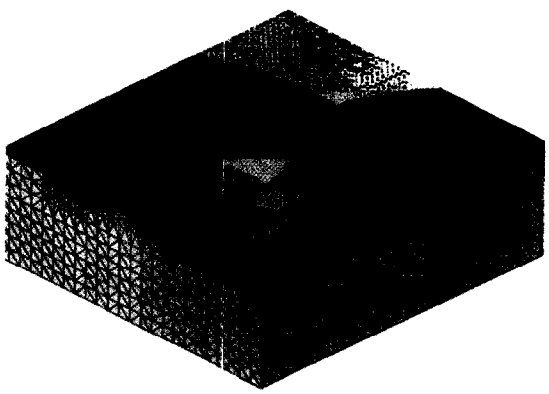

Fig. 5. Mesh partitioned for 64 subdomains.

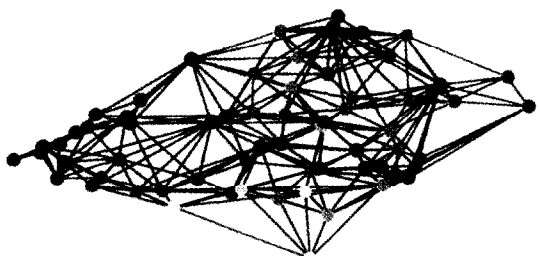

Fig. 6. Communication graph for the partitioned element mesh depicted in Fig. 5.

Table 1
Source characteristics

Date	01/21/1994
Time	18:53:44.0
Epicenter	Latitude 34°.32, Longitude −118°.48
Depth	13 km
Strike	−69°
Dip	44°
Rake	70°
Seismic moment	$M_0 = 2.4 \times 10^{22}$ dyne-cm
Rise time	$T_0 = 0.6$ s

Source function

$$\dot{M} = \begin{cases} \frac{M_0}{T_0}\left[1 - \cos\left(\frac{2\pi t}{T_0}\right)\right] & \text{if } 0 \leq t \leq T_0 \\ 0 & \text{otherwise} \end{cases}$$

Pittsburgh Supercomputing Center (PSC). The simulated time is 40 s, with a time step of 0.002 s. The simulated seismic event is a 1994 Northridge Earthquake aftershock, with its epicenter denoted by a white × in Fig. 2. The characteristics of the source were obtained from [23] and are listed in Table 1.

Figs. 7 and 8 show the E–W and N–S surface velocity components, respectively, along the d–d′ axis shown in Fig. 2; the color column on the left of the seismograms depicts the shear-wave velocity profile of the basin along the same axis. While it is clear that longer durations are associated with the deeper parts of the valley, it also seems that the constructive interference of surface and trapped body waves in the shallower regions of the

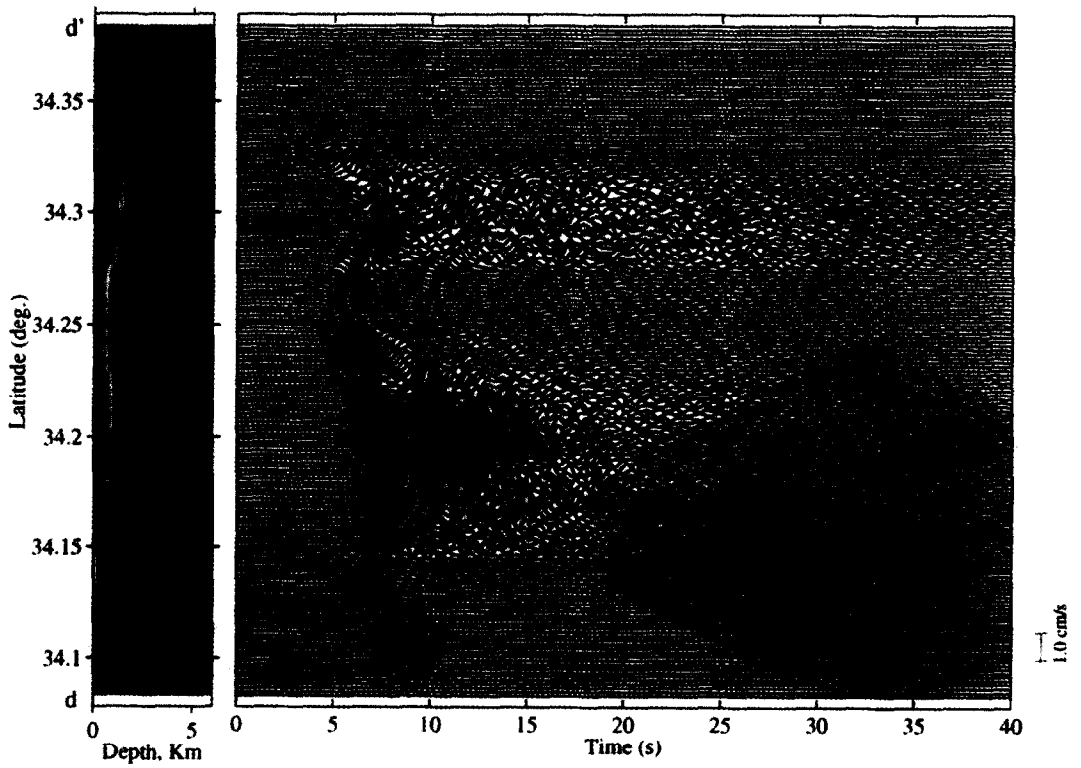

Fig. 7. Horizontal surface velocity seismogram of the E–W component along the d–d′ axis shown in Fig. 2.

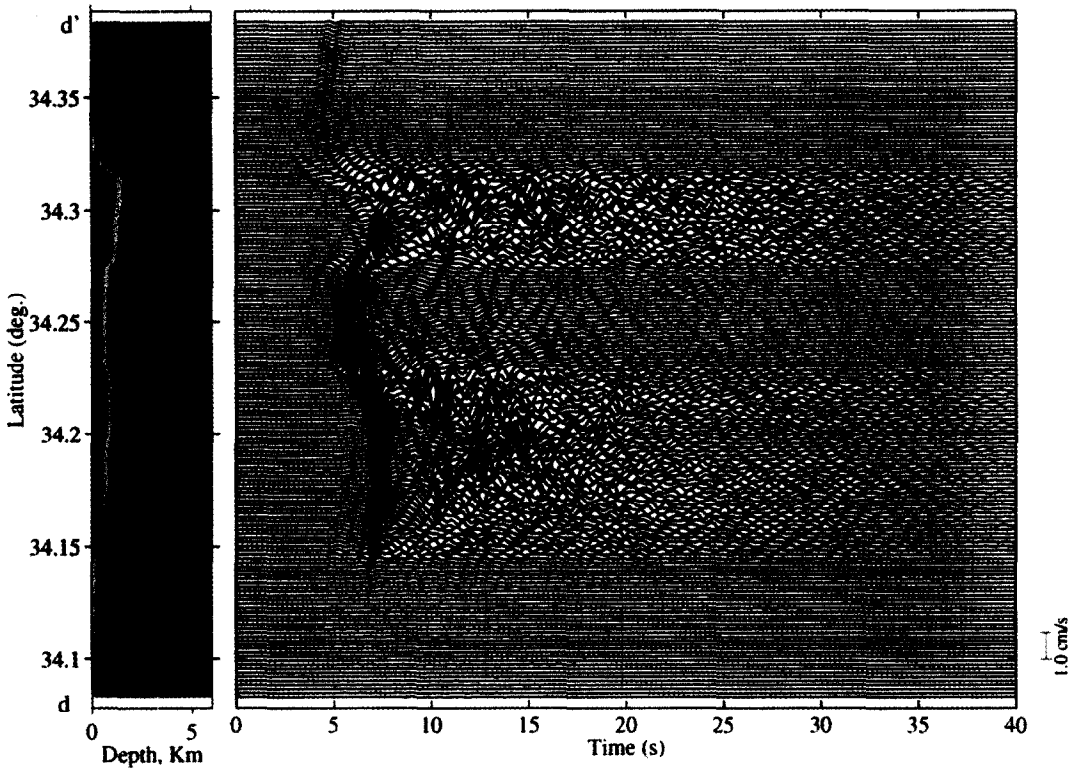

Fig. 8. Horizontal surface velocity seismogram of the N–S component along the d–d′ axis shown in Fig. 2.

valley is responsible for the stronger motion amplification observed on the surface overlying those regions. It is also noteworthy that no spurious wave reflections seem to be generated at the artificial boundaries. Fig. 9 shows the distribution of maximum surface horizontal displacements throughout the valley; it can be seen that the motion in the softer parts of the basin is amplified five to six times when compared with the motion on rock. Naturally, this is suggestive of greater damage in these regions. The solid dark line in the same figure is the outline of key topological features of the valley. By comparing the distribution of the shear-wave velocity depicted in Fig. 2 with the response shown in Fig. 9, the correlation between stronger amplification and softer layers becomes even clearer. Notice also how well the response distribution follows even the finest of the topological features; as expected, stronger response is also concentrated along different material interfaces within the valley itself.

Once the response in the time domain is obtained from the simulation, the record at every point in the valley in the frequency domain can be obtained through Fast Fourier Transforms. In Fig. 10 we plot the distribution of the amplitude of one such Fourier transform for the E–W component of the surface displacement and for a fixed frequency of 1.45 Hz. This is helpful for assessing the response of the valley at that frequency and identifying resonant regions. Indeed, the narrow stripes depicted in Fig. 10 are indicative of strong modal response; this is a property of the geological structure, which is expected to be nearly independent of the particular seismic scenario.

It is important for the design process to be able to assess the response of a hypothetical structure to a given seismic event. To this end we construct response spectra for two distinct single-degree-of-freedom oscillators. As an example, we place a simple oscillator oriented along the E–W direction at every point on the surface of the valley. We assume a natural frequency f_n for each oscillator and 5% critical damping. The valley's response along the same E–W direction is used as the excitation for the oscillator; we obtain its response in the time domain and plot the maximum relative displacement at every point in the valley. Figs. 11 and 12 depict the response spectra for two distinct oscillators with natural frequencies of 0.3 Hz and 1.45 Hz. The figures clearly assist in identifying regions where the oscillators will experience large responses; we note that the chosen values of natural frequencies are typical of tall (30-story) to moderately short (6-story) building structures.

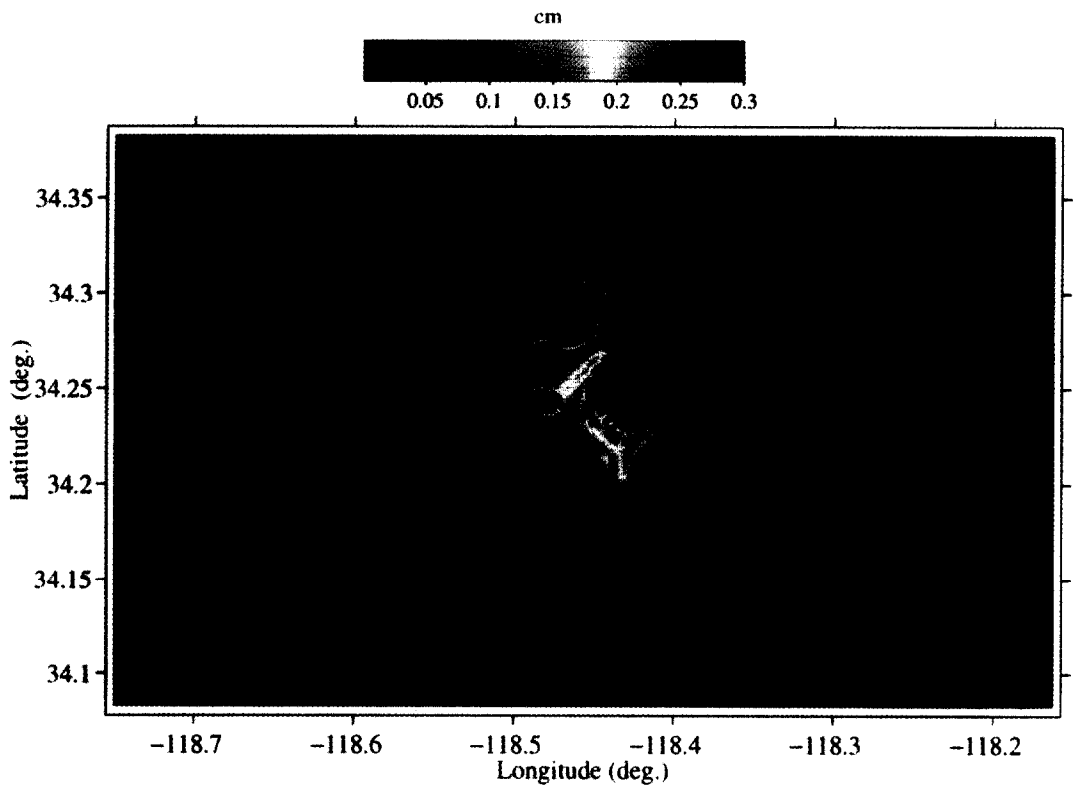

Fig. 9. Distribution of maximum horizontal surface displacement.

3.3. Performance on Cray T3D

We next discuss the performance of our parallel explicit wave propagation code on the Cray T3D. The relevant scenario for assessing the performance of our earthquake simulations as the number of processors increases is one in which the problem size increases proportionally, because unstructured PDE problems are typically memory-bound rather than compute-bound. Given a certain number of processors, we typically aim at full use of their memory; as the number of processors increases, we take advantage of their additional memory by increasing the problem size. In order to study the performance of the code with increasing problem size, we have generated a sequence of increasingly-finer meshes for the San Fernando Basin. These meshes are labeled `sf10`, `sf5`, `sf2` and `sf1`, and correspond to earthquake excitation periods of 10, 5, 2 and 1 s, respectively, and a minimum shear wave velocity of 500 m/s. Additionally, the mesh `sf1b` corresponds to the geological model used for the simulations described in the preceding section, which includes much softer soil in the top 30 m, and thus necessitates an even finer mesh. Note that mesh resolution varies with the inverse cube of the excitation period, so that halving the period results in a factor of eight increase in the number of nodes. Characteristics of the five meshes are given in Table 2.

Our timings include computation and communication but exclude I/O. We exclude I/O time because in our current implementation it is serial and unoptimized, and because the T3D has a slow I/O system. I/O time involves the time at the beginning of the program to load and read the input file produced by the parceling operation, as well as the time to output results every 30th time step to disk. With the availability of the Cray T3E at PSC, we plan to address parallel I/O in the future.

We begin with a traditional speedup histogram, for which the problem size is fixed and the number of processors is increased. Fig. 13 shows the total time, as well as the relative time spent for communication and computation, for an earthquake ground motion simulation, as a function of the number of processors. The mesh used for these timings is `sf2`. On 16 processors, the time spent for communication is 5% of the time spent for computation, which is quite good for such a highly irregular problem. There are about 24 000 nodes per

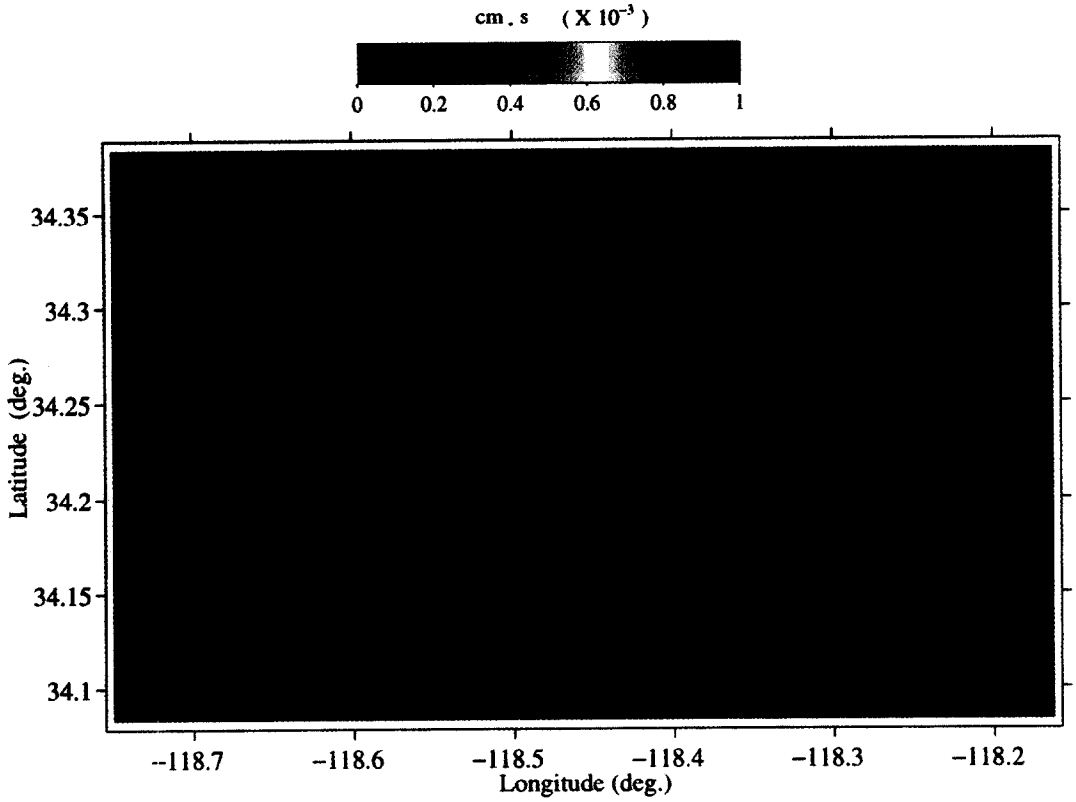

Fig. 10. Surface distribution of the amplitude of the Fourier Transform of the E–W displacement component for a frequency of 1.45 Hz.

processor, which results in about half the memory on each processor being used. As the number of processors doubles, the percentage of time spent communicating relative to computing increases, as expected. For 128 processors, the communication time has increased to one-fifth of the total time. However, we are only utilizing 1/16 of the local memory on a processor; practical simulations will generally exhibit performance more like the left bar of Fig. 13.

We can quantify the decrease in computation to communication ratio for a regular $N^{1/3} \times N^{1/3} \times N^{1/3}$ mesh. Suppose there are N/P nodes on a processor, where P is the number of processors. Suppose further that the regular grid is partitioned into cubic subdomains of equal size, one to a processor. Since computation for an explicit method such as Eq. (7) is proportional to the volume of nodes in a cube (subdomain), and communication is proportional to the number of nodes on the surface of the cube, the computation to communication ratio is proportional to $(N/P)^{1/3}$, i.e. the ratio of total nodes to surface nodes of the cube. Thus, for fixed N, the ratio is inversely proportional to $P^{1/3}$, at least for cubically-partitioned regular grids with large enough numbers of nodes per processor. Clearly, it is in our interest to keep N/P as large as possible, if we want to minimize communication time.

Consider now the case of unstructured, rather than regular, meshes. Suppose that N/P remains constant for increasing N and P, i.e. the number of nodes per processor remains constant. Now suppose that we have a partitioner that guarantees that the number of interface nodes remains roughly constant as N and P increase proportionally. Then we can expect that the computation to communication ratio will remain constant as the problem size increases.[2] In this case, we have a method that scales linearly: the amount of time required to solve a problem that is doubled in size is unchanged if we double the number of processors. Let us attempt to hold the number of nodes per processor roughly constant, and examine the aggregate performance of the machine as the problem size increases. It is difficult to maintain a constant value of N/P, since processors are available in

[2] To the extent that communication time is governed by the number of words communicated (as opposed to the number of messages, or to the route between communicating processors).

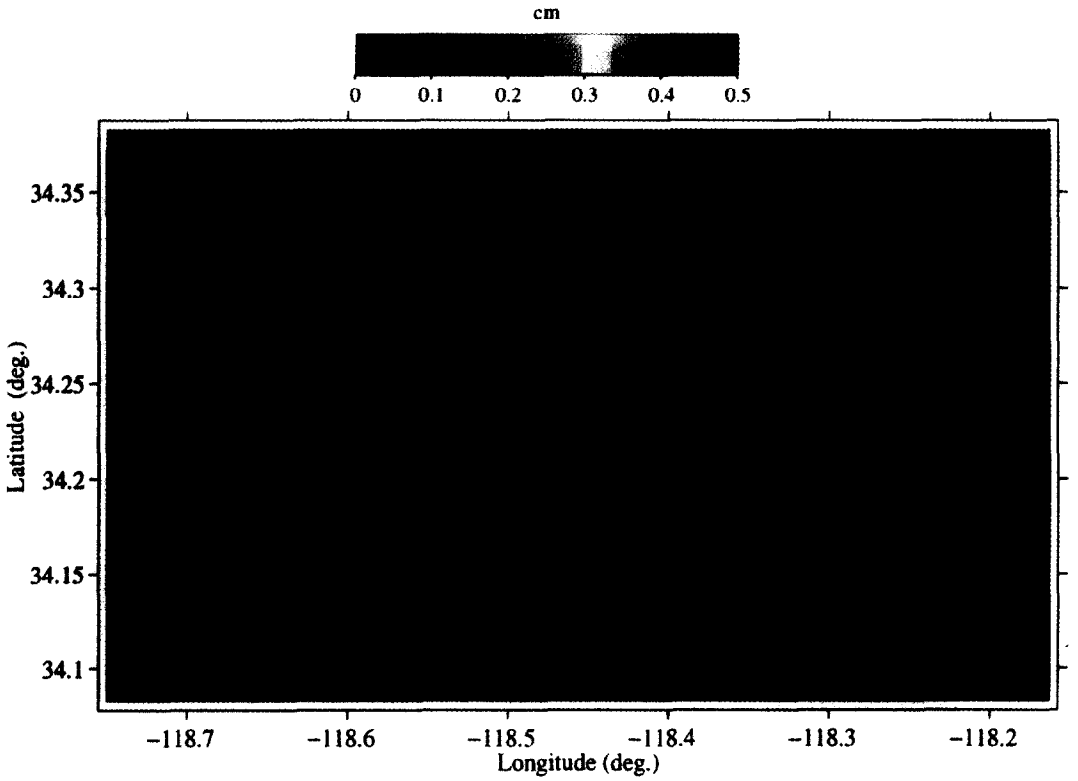

Fig. 11. Displacement response spectrum for a simple oscillator with $f_n = 0.3$ Hz and 5% critical damping.

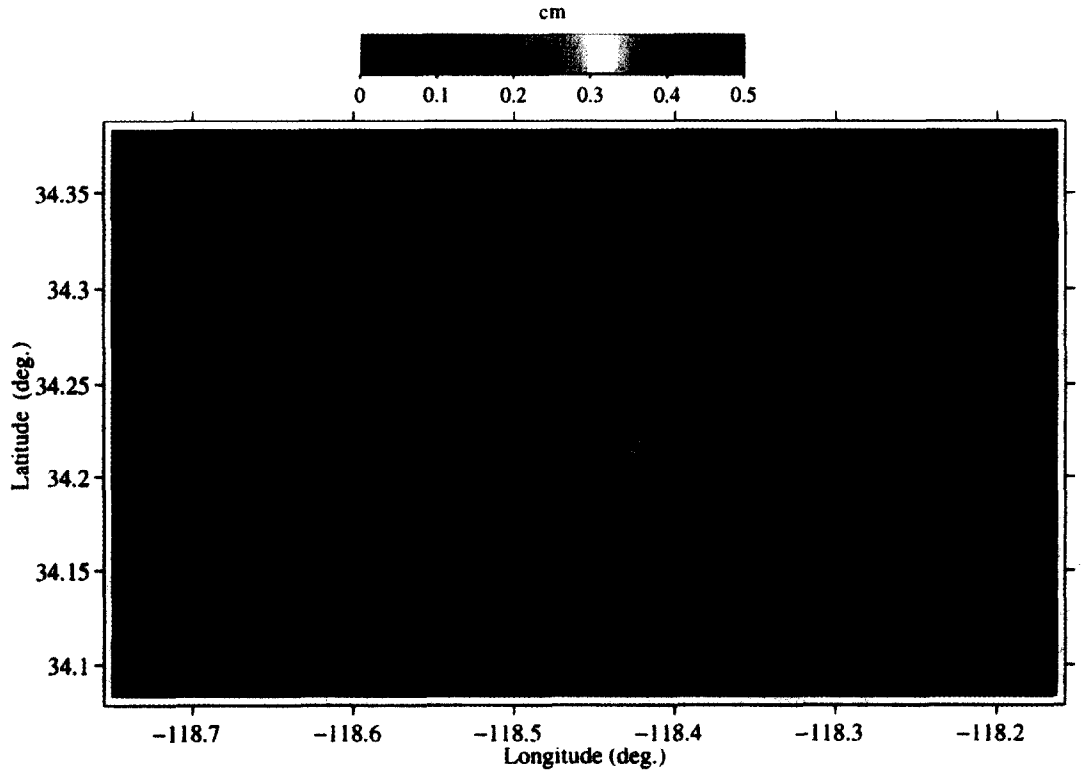

Fig. 12. Displacement response spectrum for a simple oscillator with $f_n = 1.45$ Hz and 5% critical damping.

Table 2
Characteristics of San Fernando Basin meshes

Mesh	Nodes	Equations	Elements
sf10	7 924	21 882	35 047
sf5	30 169	90 507	151 173
sf2	378 747	1 136 241	2 067 739
sf1	2 461 694	7 385 082	13 980 162
sf1b	13 422 563	40 267 689	76 778 630

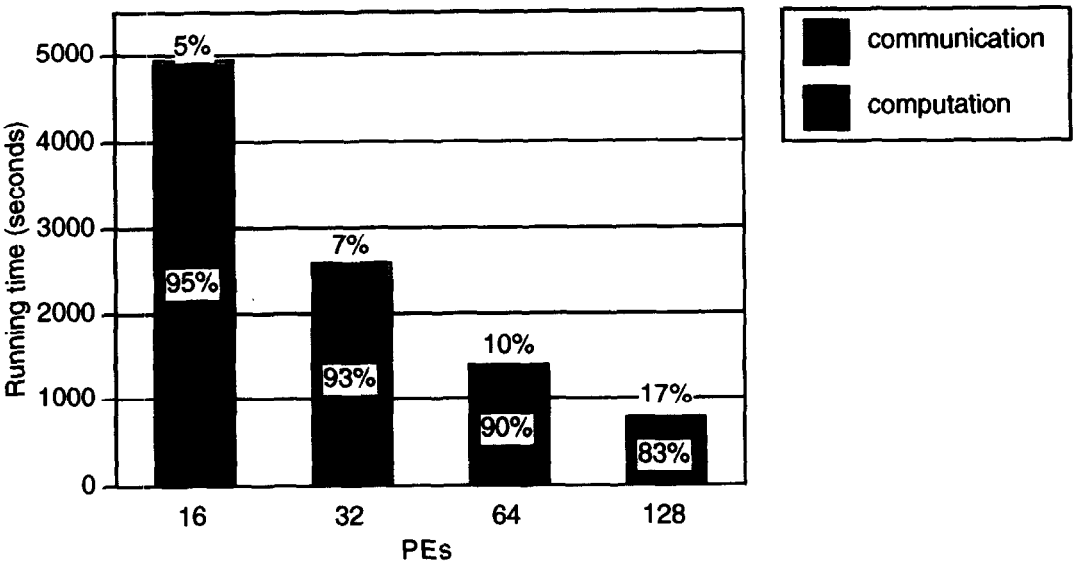

Fig. 13. Timings in seconds on a Cray T3D as a function of number of processors (PEs), excluding I/O. The breakdown of computation and communication is shown. The mesh is sf2, and 6000 time steps are carried out.

powers of two on the T3D. However, we can still draw conclusions about scalability. Fig. 14 shows the aggregate performance of our code on the T3D in megaflops per second, as a function of number of processors (and, implicitly, problem size). Megaflops are those that are sustained by matrix–vector product operations

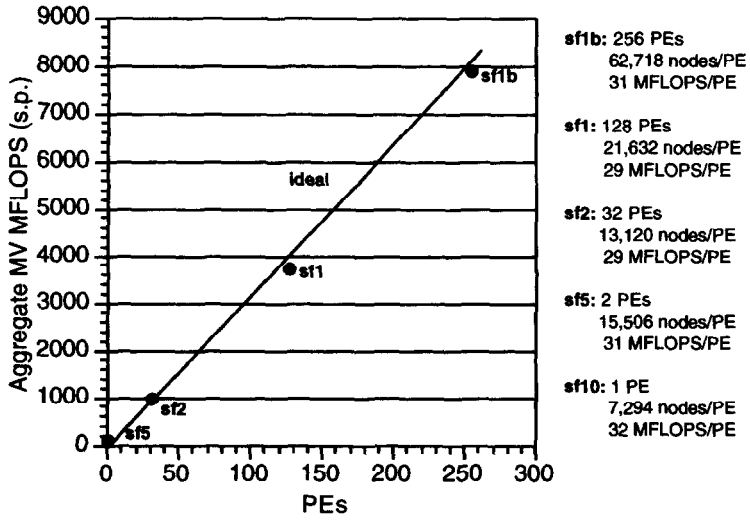

Fig. 14. Aggregate performance on Cray T3D as a function of number of processors (PEs). Rate measured for matrix–vector (MV) product operations (which account for 80% of the total running time and all of the communication) during 6000 times steps.

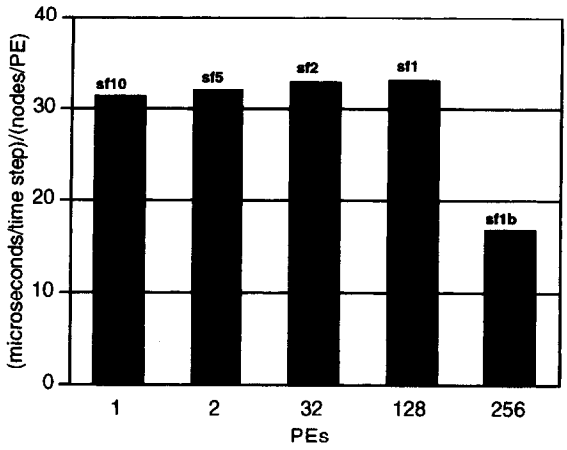

Fig. 15. T3D wall-clock time in microseconds per time step per average number of nodes per processor (PE), as a function of number of processors. This figure is based on an entire 6000 time step simulation, exclusive of I/O. The `sf1b` result is based on a damping scheme in which $\beta = 0$ in Eq. (6) so that only one matrix–vector product is performed at each time step.

(which account for 80% of the total running time and all of the communication) during a San Fernando simulation, exclusive of I/O. This figure shows nearly ideal scalability, which is defined as the single processor performance multiplied by the number of processors. These results show that excellent performance is achievable, despite the highly multiscale mesh. This behavior requires a partitioner that keeps the number of interface nodes relatively constant for problem size that increases concomitantly with number of processors.

An even better measure of scalability is to chart the time taken per time step per node. If the algorithm/implementation/hardware combination is scalable, we expect that the time taken will not change with increasing problem size. Not only must the partitioner produce 'scalable' partitions for this to happen, but in addition the PDE solver must scale linearly with N. This happens when the work per time step is $O(N)$. This is obvious from the iteration of Eq. (7)—vector sums, diagonal matrix inversions, and sparse matrix—vector multiplies require $O(N)$ operations.

Fig. 15 depicts the trend in unit wall clock time as the number of processors is increased. Unit wall clock time is measured as microseconds per time step per average number of node per processor, which includes all computations and communications for all time steps, but excludes disk I/O. As we have said above, for a truly scalable algorithm/implementation/hardware system, this number should remain constant as problem size increases with increasing processors. The figure demonstrates that we are close to this ideal. Ultimately, wall clock time per node per time step is the most meaningful measure of scalable performance for our application, since it is a direct indicator of the ability to solve our ultimate target problems, which are an order of magnitude larger than the San Fernando Basin problem we have described in this paper.

4. Concluding remarks

We have described our approach to modeling the earthquake-induced ground motion in large, heterogeneous basins on parallel computers. By paying careful attention to the impact on parallel execution of all components of the code, we are able to obtain excellent performance on highly unstructured mesh problems. In particular, through the use of (i) space- and time-localized absorbing boundaries; (ii) seismic input in the form of effective boundary or interior forces applied at the element level; (iii) explicit numerical techniques for the wave propagation problem; (iv) strict control of mesh resolution and aspect ratio; and (v) an asymptotically optimal mesh partitioner, we obtain excellent scalability of the parallel code. The Archimedes toolset integrates the basic components necessary for solving general PDE problems involving static unstructured meshes on parallel distributed memory systems. These components include meshing, partitioning, and parallel code generation.

We currently solve the meshing, partitioning, and parceling problems sequentially on a large shared-memory machine. Our ultimate target problem—the Greater Los Angeles Basin with an excitation of 2 Hz and with soil

deposits having shear wave velocities as low as 200 m/s—will require meshes on the order of hundreds of millions of elements. Despite the fact that our sequential meshing and partitioning codes are fast, we may have to parallelize these steps in order to solve the target problem, primarily for memory reasons. The scalability of the parallel portion of our code suggests that our target problem is within reach.

Acknowledgments

Special thanks are given to Keiiti Aki, Francisco J. Sanchez-Sesma, and Yoshiaki Hisada for many useful discussions. This research was supported by the National Science Foundation's Grand Challenges in High Performance Computing and Communications program, under grant CMS-9318163. Funding comes from the Directorate for Computer and Information Science and Engineering, the Directorate for Engineering, and the Directorate for Earth and Atomspheric Sciences. In addition, NSF funding was supplemented with funds from the Advanced Research Projects Agency. The cognizant NSF program official is Dr. Clifford J. Astill. Computing services on the Pittsburgh Supercomputing Center's Cray T3D and DEC 8400 were provided under PSC grant BCS-960001P. We thank Harold Magistrale and Steve Day of San Diego State University for providing the material property model of the San Fernando Valley.

References

[1] K. Aki, Local site effect on ground motion, in: J. Lawrence Von Thun, ed., Earthquake Engineering and Soil Dynamics. II: Recent Advances in Ground-Motion Evaluation (ASCE, 1988) 103–155.

[2] H. Bao, J. Bielak, O. Ghattas, L.F. Kallivokas, D.R. O'Hallaron, J.R. Shewchuk and J. Xu, Earthquake ground motion modeling on parallel computers, in: Proc. Supercomputing '96, November 1996.

[3] J.P. Bardet and C. Davis, Engineering observations on ground motions at the Van Norman Complex after the 1994 Northridge Earthquake, Bull. Seism. Soc. Am. 86(1B) (1996) S333–S349.

[4] J. Bielak and P. Christiano, On the effective seismic input for nonlinear soil-structure interaction systems, Earthq. Engrg. Struct. Dyn. 12 (1984) 107–119.

[5] A. Bowyer, Computing Dirichlet tessellations, Comput. J. 24(2) (1981) 162–166.

[6] M.G. Cremonini, P. Christiano and J. Bielak, Implementation of effective seismic input for soil-structure interaction systems, Earthq. Engrg. Struct. Dyn. 16 (1988) 615–625.

[7] A. Feldmann, O. Ghattas, J.R. Gilbert, G.L. Miller, D.R. O'Hallaron, E.J. Schwabe, J.R. Shewchuk and S.-H. Teng, Automated parallel solution of unstructured PDE problems, to appear. Available from http://www.cs.cmu.edu/afs/cs.cmu.edu/project/quake/public/papers/pde.color.ps.

[8] A. Frankel and J.E. Vidale, A three-dimensional simulation of seismic waves in the Santa Clara Valley, California from a Loma Prieta aftershock, Bull. Seism. Soc. Am. 82 (1992) 2045–2074.

[9] R.W. Graves, Modeling three-dimensional site response effects in the Marina District Basin, San Francisco, California, Bull. Seism. Soc. Am. 83 (1993) 1042–1063.

[10] S. Hartzell, A. Leeds, A. Frankel and J. Michael, Site response for urban Los Angeles using aftershocks of the Northridge earthquake, Bull. Seism. Soc. Am. 86(1B) (1996) S168–S192.

[11] H. Kawase and K. Aki, A study on the response of a soft basin for incident S, P, and Rayleigh waves with special reference to the long duration observed in Mexico City, Bull. Seism. Soc. Am. 79 (1989) 1361–1382.

[12] H.-L. Liu and T. Heaton, Array analysis of the ground velocities and accelerations from the 1971 San Fernando, California, earthquake, Bull. Seism. Soc. Am. 74 (1996) 1951–1968.

[13] H. Magistrale, K.L. McLaughlin and S.M. Day, A geology-based 3-D velocity model of the Los Angeles Basin sediments, Bull. Seism. Soc. Am. 86 (1996) 1161–1166.

[14] K.L. McLaughlin and S.M. Day, 3D elastic finite difference seismic wave simulations, Comput. Phys., Nov/Dec 1994.

[15] G.L. Miller, S.-H. Teng, W. Thurston and S.A. Vavasis, Automatic mesh partitioning, in: A. George, J. Gilbert and J. Liu, eds., Graph Theory and Sparse Matrix Computation, Vol. 56 of The IMA Volumes in Mathematics and its Application (Springer-Verlag, 1993) 57–84.

[16] K.B. Olsen and R.J. Archuleta, Three-dimensional simulation of earthquakes on the Los Angeles Fault System, Bull. Seism. Soc. Am. 86 (1996) 575–596.

[17] K.B. Olsen, R.J. Archuleta and J.R. Matarese, Magnitude 7.75 earthquake on the San Andreas fault: Three-dimensional ground motion in Los Angeles, Science 270(5242) (1995) 1628–1632.

[18] F.J. Sánchez-Sesma and F. Luzón, Seismic response of three-dimensional valleys for incident P, S, and Rayleigh waves, Bull. Seism. Soc. Am. 85 (1995) 269–284.

[19] P. Spudich and M. Iida, The seismic coda, side effects, and scattering in alluvial basins studied using aftershocks of the 1986 North Palm Springs, California earthquake as source arrays, Bull. Seism. Soc. Am. 83 (1993) 1721–1724.

[20] J.R. Shewchuk, Robust adaptive floating-point geometric predicates, in: Proc. Twelfth Annual Symposium on Computational Geometry. Association for Computing Machinery (May 1996) 141–150.

[21] J.R. Shewchuk, Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator, in: First Workshop on Applied Computational Geometry, Association for Computing Machinery (May 1996) 124–133.

[22] J.R. Shewchuk and O. Ghattas, A compiler for parallel finite element methods with domain-decomposed unstructured meshes, in: D.E. Keyes and J. Xu, eds., Domain Decomposition Methods in Scientific and Engineering Computing, Vol. 180 of Contemporary Mathematics (American Mathematical Society, 1994) 445–450.

[23] H.K. Thio and H. Kanamori, Source complexity of the 1994 Northridge Earthquake and its relation to aftershock mechanisms, Bull. Seism. Soc. Am. 86(1B) (1996) S84–S92.

[24] J.E. Vidale and D.V. Helmberger, Elastic finite-difference modeling of the 1971 San Fernando, California, earthquake, Bull. Seism. Soc. Am. 78 (1988) 122–141.

[25] D.F. Watson, Computing the n-dimensional Delaunay tessellation with application to Voronoï polytopes, Comput. J. 24(2) (1981) 167–172.