USE OF HIGH-PERFORMANCE COMPUTERS, FEA AND THE CAVE AUTOMATIC VIRTUAL ENVIRONMENT FOR COLLABORATIVE DESIGN OF COMPLEX SYSTEMS

by

Edward J. Plaskacz
Computational Scientist
Argonne National Laboratory

Ronald F. Kulak
Manager, Computational Mechanics Section
Argonne National Laboratory

ABSTRACT:

Concurrent, interactive engineering design and analysis has the potential for substantially reducing product development time and enhancing US competitiveness. Traditionally, engineering design has involved running engineering analysis codes to simulate and evaluate the response of a product or process, writing the output data to file, and viewing or "post-processing" the results at a later time. The emergence of high-performance computer architectures, virtual reality, and advanced telecommunications in the mid 90's promises to dramatically alter the way designers, manufacturers, engineers and scientists will do their work.

INTRODUCTION:

The iterative nature of engineering design is governed in time by the cycle of design and analysis. Design engineers are often faced with the prospect of changing a design to simultaneously satisfy constraints imposed by manufacturability, serviceability, and safety. Traditionally, the large blocks of time required for testing each combination of parameters has restricted the number of simulations which could be performed before the product deadlines were impacted. This, in turn, restricts productivity, optimality, and in most cases, creativity. High-performance computer architectures have proven themselves to be an effective avenue for bridging the gap between computational needs and the power of computational hardware in many disciplines. Our experience with message-passing in the context of nonlinear explicit transient finite element simulations on the 512 processor Intel Delta resulted in a speedup of 333 relative to a uniprocessor run. The practical significance of a two orders-of-magnitude increase in computational speed is that it enables designers and engineers to perform more simulations and hence find better designs. In addition, large-scale detailed simulations become manageable and the need to break up the problem and model each region separately is reduced. High-performance computers are shifting engineering design bottlenecks away from simulation turn-around time and towards information sharing between the experts.

This paper presents an overview of our parallel algorithms targeted to bring the power of high-performance computer architectures to bear on finite element
DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.
USE OF HIGH-PERFORMANCE COMPUTERS, FEA AND THE CAVE AUTOMATIC VIRTUAL ENVIRONMENT FOR COLLABORATIVE DESIGN OF COMPLEX SYSTEMS
by Edward J. Plaskacz and Ronald F. Kulak

analysis. It continues with a description of our work with the CAVE Virtual Reality Environment which facilitate its use as a mechanism for effective collaborations between participants who are geographically distributed. True collaborations involve ongoing communication. Traditionally, for geographically distributed participants, this has involved traveling. Teleconferencing has reduced this burden somewhat but nevertheless remains inadequate due to its lack of ubiquity. State-of-the-art virtual reality such as the CAVE promises the ability to bring researchers together through telepresence, to take tours through the facilities of collaborators and to take virtual tours through a simulation with ideas being exchanged.

Among the possible applications for these new technologies is the paradigm of concurrent engineering wherein the design, analysis, redesign, manufacturing feasibility, and overall optimization is addressed concurrently.

PARALLEL FINITE ELEMENT FORMULATION:

This section summarizes the essential ingredients for parallel nonlinear finite element analysis. The first subsection presents an overview of the domain decomposition step, which is used to subdivide the complete finite element mesh into pieces analyzed on separate processors. The second subsection discusses the p4 message-passing package, which is the software that allows parallel processors to communicate with each other. The next subsection summarizes previous work in the context of nonlinear explicit transient finite element simulations. The final subsection summarizes our parallel dynamic relaxation algorithm for the solution of nonlinear static problems in structural mechanics.

Domain Decomposition

When finite element analysis is performed on either traditional sequential computers or vector computers, one finite element mesh is developed to represent the physical problem under study. In contrast, since each processor in a parallel computing platform only operates on part of the full problem, it requires the portion of data on which it operates. A process called domain decomposition is used to subdivide the finite element grid into \( N_p \) grids, where \( N_p \) is the number of processors that will be used for computation. This process is illustrated in Fig. 1. The upper part of the figure shows a complete finite element mesh (i.e., the computational domain), and the lower part shows how the mesh could be subdivided (i.e., decomposed) into three submeshes for parallel computing using three processors. It should be noted that elements can belong only to one subdomain, but nodes may belong to multiple subdomains. Thus, nodal information must be exchanged between domains, which is discussed in the next subsection.
Many methods have been proposed for doing domain decomposition. Farhat [1] has proposed a simple and effective method for doing static domain decomposition. Al-Nasra and Nguyen [2] also described a method applicable to finite element analysis.

We incorporated a combination of the above algorithms and code into the preprocessor stage of our finite element procedure. For parallel computing, the first step is to generate input data using either in-house or commercial mesh generation software. This is also the first step when doing sequential or vector computing. The second step is to process the original input data set through the domain decomposer and produce $N_p$ input data sets, one for each processor.

---

**Figure 1. An Example of Domain Decomposition**
USE OF HIGH-PERFORMANCE COMPUTERS, FEA AND THE CAVE AUTOMATIC VIRTUAL ENVIRONMENT FOR COLLABORATIVE DESIGN OF COMPLEX SYSTEMS
by Edward J. Plaskacz and Ronald F. Kulak

Message Passing

The nodes that lie completely within a domain have the forces available for the solution of the equations of motion. However, the forces at the interfaces of contiguous domains must be exchanged before the equations can be solved. Note, this leads to redundant calculations of the equations of motion for the interface nodes; however, this is not significant.

Message passing was selected here to exchange information among processors because of portability concerns. The Argonne-developed p4 message-passing system [3] [4] provides a powerful tool for the adaptation of finite element codes to a wide range of parallel platforms.

Parallel Explicit Transient Finite Element Algorithm

The evolution of a finite element program with explicit time integration from a sequential form to a parallel message-passing incarnation is presented in [5]. Here we present an outline of the basic theory and the parallel algorithm.

After spatial semi-discretization is performed, the governing equations of motion are:

\[ \mathbf{M} \ddot{\mathbf{u}}(t) = \mathbf{f}(t), \text{ where } \mathbf{f}(t) = \mathbf{f}^{\text{ext}}(t) - \mathbf{f}^{\text{int}}(t). \]  

(1)

The equation of motion is integrated in time using the central difference algorithm. The updates are given by

\[ \dot{\mathbf{u}}(t) = \mathbf{M}^{-1} \mathbf{f}(t) \]

\[ \dot{\mathbf{u}} \left( t + \frac{1}{2} \Delta t \right) = \dot{\mathbf{u}} \left( t - \frac{1}{2} \Delta t \right) + \ddot{\mathbf{u}}(t) \Delta t \]

\[ \mathbf{u}(t + \Delta t) = \mathbf{u}(t) + \dot{\mathbf{u}} \left( t + \frac{1}{2} \Delta t \right) \Delta t \]

(2)

Table 1 outlines the computational flow for our parallel explicit central difference algorithm.
USE OF HIGH-PERFORMANCE COMPUTERS, FEA AND THE CAVE AUTOMATIC VIRTUAL ENVIRONMENT FOR COLLABORATIVE DESIGN OF COMPLEX SYSTEMS
by Edward J. Plaskacz and Ronald F. Kulak

Message Passing

The nodes that lie completely within a domain have the forces available for the solution of the equations of motion. However, the forces at the interfaces of contiguous domains must be exchanged before the equations can be solved. Note, this leads to redundant calculations of the equations of motion for the interface nodes; however, this is not significant.

Message passing was selected here to exchange information among processors because of portability concerns. The Argonne-developed p4 message-passing system [3] [4] provides a powerful tool for the adaptation of finite element codes to a wide range of parallel platforms.

Parallel Explicit Transient Finite Element Algorithm

The evolution of a finite element program with explicit time integration from a sequential form to a parallel message-passing incarnation is presented in [5]. Here we present an outline of the basic theory and the parallel algorithm.

After spatial semi-discretization is performed, the governing equations of motion are:

\[ M \ddot{u}(t) = f(t), \text{ where } f(t) = f^{xt}(t) - f^{int}(t). \]  

The equation of motion is integrated in time using the central difference algorithm. The updates are given by

\[ \ddot{u}(t) = M^{-1} f(t) \]

\[ \ddot{u}\left(t + \frac{1}{2} \Delta t\right) = \ddot{u}\left(t - \frac{1}{2} \Delta t\right) + \ddot{u}(t) \Delta t \]  

\[ u(t + \Delta t) = u(t) + \dot{u}\left(t + \frac{1}{2} \Delta t\right) \Delta t \]  

Table 1 outlines the computational flow for our parallel explicit central difference algorithm.
Table 1. Flowchart for Explicit Integration in a Parallel Distributed Environment

1. Perform domain decomposition
2. Initialize: $u(0)$, $\dot{u}(t)$, $t = 0$
3. Parallel Central Difference
   A. Loop over elements in each processor, $p$: $e = 1$ to $n_e$
      a. Compute $f_{\text{int}}(t)$
      b. Compute $f_{\text{ext}}(t)$
      End loop over elements
   B. Exchange interface nodal forces, $f_{\text{int}}(t)$ and $f_{\text{ext}}(t)$, between contiguous processors
   C. Loop over nodes in each processor, $p$: $i = 1$ to $n_n$
      a. Compute $\ddot{u}_i(t)$
      b. Compute $\dot{u}_i\left(t + \frac{1}{2}\Delta t\right)$
      c. Compute $u_i(t + \Delta t)$
      End loop over nodes
   D. Check for last step:
      If yes, stop
      If no, $t = t + \Delta t$, go to A

Parallel Dynamic Relaxation Algorithm

The static solution of three-dimensional nonlinear problems requires methods that are robust and efficient. Dynamic Relaxation (DR) is an explicit method that is extremely robust when applied to highly nonlinear problems. Since DR falls into the explicit class of integrators, it does not require the solution of a set of simultaneous equations. Instead, each equation can be solved individually at each step once the internal and external forces are known. Although it may not always be the most efficient method for all problems, DR almost always obtains the solution. The first papers on dynamic relaxation were published by Day [6] and Otter [7].

The work presented here summarizes the parallelization of the DR algorithm presented by Kulak, Plaskacz and Pfeiffer [7], which was based on the sequential algorithm presented by Underwood [8]. In our approach to DR, a Rayleigh quotient is used to determine optimum damping. The two summations that are used to compute the quotient are obtained by adding the partial sums from each subdomain. These partial sums must be exchanged between processors. Similarly, several quantities used for the convergence calculations must be exchanged among processors.
With the DR method, the solution of the static equilibrium equations are obtained as the solution to the dynamic equilibrium equations that include damping. Thus, the semidiscritized equations of motion are given by Eq. (1) above.

The central difference integrator is used to solve the equations of motion, but a modified update is used for the velocity, \( \dot{\mathbf{u}} \),

\[
\dot{\mathbf{u}} = \alpha_1 \frac{1}{2} (\dot{\mathbf{u}} - \dot{\mathbf{u}}_{t-1}) + \alpha_2 \Delta t \dot{\mathbf{u}}_t,
\]

where \( \Delta t \) is a fictitious time increment and \( n \) is the step number. Two parameters, \( \alpha_1 \) and \( \alpha_2 \), appear in the update for the velocity and they control the damping that is put into the system. The following forms are used to compute \( \alpha_1 \) and \( \alpha_2 \),

\[
\alpha_1 = \frac{2 - c \Delta t}{2 + c \Delta t}, \quad \alpha_2 = \frac{2}{2 + c \Delta t}, \quad c = 2 \omega_i
\]

where \( c \) is the damping and \( \omega_i \) is the lowest participating frequency (Underwood, [8]) of the system. The lowest participating frequency is calculated from the following approximate Rayleigh quotient

\[
\omega_i^2 = \frac{\mathbf{u}^T \mathbf{K}_{\text{diag}} \mathbf{u}}{\mathbf{u}^T \mathbf{M}_{\text{diag}} \mathbf{u}}
\]

where the approximate diagonal stiffness matrix, \( \mathbf{K}_{\text{diag}} \), has the diagonal terms, \( K_{ii}^a \), computed from

\[
K_{ii}^a = \left[ f_{i,t}^{\text{int},t} - f_{i,t}^{\text{int},t-1} \right] \left[ \Delta t \dot{u}_{i,t-1}^{t-1} \Delta t \right].
\]

Since the DR algorithm is an iterative solution procedure, it is necessary to use a convergence criterion to determine when the solution vector is close enough to the true solution. In this work, a dual criterion has been found to be effective and efficient. The criteria are

\[
\frac{||f_{i,t}^{\text{ext},t} - f_{i,t}^{\text{int},t}||}{||f_{i,t}^{\text{ext},t}||} \times 100 \leq \epsilon_f, \quad \frac{||\dot{u}_{i,t}^{t-1}||}{||\dot{u}_{i,t-1}^{t-1}||} \times 100 \leq \epsilon_u,
\]

where \( || \cdot ||_2 \) denotes the Euclidean norm. For engineering analysis we have found that \( \epsilon_f = 1.0 \) and \( \epsilon_u = 0.1 \) give a good balance between accuracy and efficiency.
USE OF HIGH-PERFORMANCE COMPUTERS, FEA AND THE CAVE AUTOMATIC VIRTUAL ENVIRONMENT FOR COLLABORATIVE DESIGN OF COMPLEX SYSTEMS
by Edward J. Plaskacz and Ronald F. Kulak

Since element size may vary within the computational mesh and reduce the convergence rate, the element densities, which are a free parameter, have been chosen so that the transient times for all elements is equal to one, i.e., $\Delta t = 1.0$.

Table 2 shows the computational flow for our parallelized DR algorithm.

Table 2. Flowchart for Parallel Dynamic Relaxation Algorithm

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Perform domain decomposition</td>
</tr>
<tr>
<td>2.</td>
<td>Initialize: $u(0)$, $\dot{u}(0)$, $t = 0$, $N = 1$</td>
</tr>
<tr>
<td>3.</td>
<td>Calculate fictitious element densities and nodal masses</td>
</tr>
<tr>
<td>4.</td>
<td>Parallel Dynamic Relaxation</td>
</tr>
<tr>
<td>A.</td>
<td>Loop over elements in each processor: $e = 1$ to $n_e$</td>
</tr>
<tr>
<td>a.</td>
<td>Compute $f^{int}$</td>
</tr>
<tr>
<td>b.</td>
<td>Compute $f^{ext}$</td>
</tr>
<tr>
<td></td>
<td>End loop over elements</td>
</tr>
<tr>
<td>B.</td>
<td>SEND/RECEIVE interface nodal forces, $f^{int}$ and $f^{ext}$, between contiguous processors</td>
</tr>
<tr>
<td>C.</td>
<td>Loop over nodes in each processor: $I = 1$ to $n_n$</td>
</tr>
<tr>
<td>a.</td>
<td>Compute $K_{II}$ and partial Rayleigh sums</td>
</tr>
<tr>
<td>b.</td>
<td>SEND/RECEIVE subdomain partial Rayleigh sums among all processors</td>
</tr>
<tr>
<td>c.</td>
<td>Compute $\omega_1$, $c$, $\alpha_1$, $\alpha_2$</td>
</tr>
<tr>
<td>d.</td>
<td>Compute $\bar{u}$, $\bar{\dot{u}}$, $\bar{\ddot{u}}$</td>
</tr>
<tr>
<td>e.</td>
<td>SEND/RECEIVE subdomain convergence data</td>
</tr>
<tr>
<td>f.</td>
<td>Check for step convergence:</td>
</tr>
<tr>
<td></td>
<td>If yes, go to next load step: $t = 0.0$; $N = N+1$; $u(0) = 0$; $\dot{u}(0) = 0$</td>
</tr>
<tr>
<td></td>
<td>If no, go to next iteration: $t = t + \Delta t$</td>
</tr>
<tr>
<td>g.</td>
<td>Check for last load step:</td>
</tr>
<tr>
<td></td>
<td>If yes, stop</td>
</tr>
<tr>
<td></td>
<td>If no, go to A</td>
</tr>
</tbody>
</table>

VISUALIZATION:

Through the use of the post-processing software, Virtual Reality visualization (VRviz) [10], and the Cave Automatic Virtual Environment (CAVE), finite element representations can be viewed in three dimensions. Unlike a representation on a two-dimensional screen, the CAVE does not allow for misrepresentation of proximities.
VRviz has several visualization options. A user is able to translate the image, rotate it about any axis or rescale it at any time. The simulation results may be replayed in slow motion, viewed frame-by-frame or frozen at any time step. These capabilities allow the user to navigate around, or even inside, the image in order to effectively analyze possible problem areas and redesign as necessary.

The CAVE’s utility is not restricted to providing lone researchers with a three-dimensional visualization capability. The CAVE’s strongest feature is that it provides and promotes collaborative engineering. A single CAVE can be used by a group of experts to take a virtual tour through a simulation. Each expert can participate in the redesign of a component in the presence of other experts who can assess the impact of the proposed change on other aspects of component behavior. Engineered structures as commonplace as buildings and bridges as well as more esoteric structures such as aerospace planes can be more optimally designed given a more collaborative environment. For example, a design change made to improve vehicle crashworthiness can be immediately assessed by a noise and vibration specialist without any intermediaries, paperwork, or time delay. By linking multiple CAVES together by high-speed networks, collaboration among geographically distributed partners is possible.

The CAVE-to-CAVE software library is a collection of functions designed to facilitate shared visualization experiences among remote virtual environments anywhere on the internet. The CAVE-to-CAVE library establishes three processes. The broker process constructs and maintains a database containing information describing the session participants. The server process first informs the broker of its existence by registering with the broker. Next, the server process registers the information stream available for the client processes to subscribe to. To minimize the amount of information transmitted over the internet (for both speed and security purposes), the finite element data is assumed to reside locally at each participant’s CAVE. Only nine parameters need to be broadcast to synchronize the displays in all CAVES participating in a session. These nine parameters are: FrameId, RunStatus, ZoomStatus, TransXStatus, TransYStatus, TransZStatus, RotXStatus, RotYStatus, and RotZStatus. Client processes register with the broker, subscribe to an information stream and execute a callback function each time a message is received. In the case of VRviz, a function named DrawData uses the parameters broadcast from the server process and the locally stored finite element data to display the same image in each of the CAVES controlled by server processes. Figure 2 is a schematic representation of a shared visualization of an impact-contact calculation between two remote CAVES. To present the “illusion” that the users are all in the same room, each remote user is represented as a stick figure. In the future, these crude stick figure images will be replaced with more true-to-life representations.
USE OF HIGH-PERFORMANCE COMPUTERS, FEA AND THE CAVE AUTOMATIC VIRTUAL ENVIRONMENT FOR COLLABORATIVE DESIGN OF COMPLEX SYSTEMS
by Edward J. Plaskacz and Ronald F. Kulak

Figure 2. VRviz Application Being Displayed in Multiple CAVEs

ACKNOWLEDGMENTS:

This work was performed at Argonne National Laboratory, a contract laboratory of the United States Department of Energy under contract W-31-109-Eng-38.

REFERENCES:


USE OF HIGH-PERFORMANCE COMPUTERS, FEA AND THE CAVE AUTOMATIC VIRTUAL ENVIRONMENT FOR COLLABORATIVE DESIGN OF COMPLEX SYSTEMS
by Edward J. Plaskacz and Ronald F. Kulak


DISCLAIMER

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